Universidad
Carlos III de Madrid

Departamento de Tecnología Electrónica

PROYECTO FIN DE CARRERA

Estudio de la viabilidad de un sistema software de inyección de fallos para un microprocesador Cortex-M1 empotrado en FPGA

Autor: Fco Javier Pérez Sanjurjo

Tutor: Marta Portela García

Leganés, Octubre de 2012

Título: Estudio de la viabilidad de un sistema software de inyección de fallos para un microprocesador Cortex-M1 empotrado en FPGA

Autor: Fco Javier Pérez Sanjurjo

Director: Marta Portela García

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco en primer lugar a mi familia (Mamá, Papá, Ana y Carlitos) el apoyo que me han dado en la toma de mis decisiones así como el ánimo suficiente en momentos malos para proseguir con mis objetivos.

Debo agradecer también a todos aquellos que empezaron siendo simples compañeros de facultad y se han convertido en íntimos amigos, por estos inolvidables años juntos. Elisa, Elvira, Andrés, Edu, David, Javier, Víctor, Luis, Ricardo, Pablos, Carlos y Alejandro.

Por último, me gustaría agradecer a los profesores de la universidad, sin los cuales, no habría podido acceder a mi actual situación laboral. Basam, Carlos Marcos y muy en especial a Marta Portela García.

Resumen

Este proyecto trata del estudio de la viabilidad de inyección de fallos programada en un sistema constituido por un microprocesador Cortex-M1 empotrado en una FPGA de Actel.

En el proyecto se estudian los conceptos necesarios en cuanto a la arquitectura del microprocesador y de la placa de pruebas de Actel, así como de las herramientas de ayuda para diseñar e implementar el sistema empotrado (componentes hardware y software) y realizar la depuración del mismo. Además, aporta tutoriales de ayuda en pasos que pueden resultar difíciles o en los cuales no hay información suficiente.

Los resultados de las pruebas realizadas muestran cómo habría que proceder para realizar una inyección de fallos y qué herramientas serían necesarias.

Palabras clave: FPGA, arquitectura Cortex M-1, sistema de inyección de fallos, tutorial Libero Soc, tutorial SoftConsole.

Abstract

This project studies the feasibility of performing fault injection in a microprocessor-based system that consists in aCortex-M1 core embedded in a FPGA from Actel.

The project presents the necessary concepts regarding the architecture of the microprocessor and the evaluation FPGA board from Actel, as well as, the software tools used to design and implement the embedded system (hardware and software components) and to perform debugging. In addition, helpful tutorials are included in order to guide on steps that can be difficult or don't have enough information.

The results of the tests show how should proceed to make a fault injection and what tools would be needed.

Keywords: FPGA, architecture of Cortex M-1, fault injection system, tutorial of Libero Soc, tutorial of SoftConsole.

Índice general

Índice de figuras

Capítulo 1

Introducción y objetivos

1.1 Introducción

Hoy en día vivimos en un mundo rodeado de productos electrónicos digitales. No podemos mirar en ninguna dirección sin encontrar un objeto que posea algún circuito digital. Móviles, ordenadores, relojes, automóviles, son solo algunos de los claros ejemplos que muestran el aumento de circuitos digitales en nuestro entorno. Esto es debido a la mejora de prestaciones que han aportado en las aplicaciones antes mencionadas, como el aumento de capacidad de procesamiento de información, la fiabilidad o el descenso del consumo eléctrico. El aumento de componentes electrónicos digitales se ha dado en todos los ámbitos, aunque pudiera empezar en ámbitos específicos, como la industria armamentística, actualmente se encuentran incluidos hasta en el sector domestico. De manera que la dependencia en este tipo de producto ha aumentado con los años.

Como consecuencia de esto, en ciertas aplicaciones, la importancia en el control de fallos es muy elevada, como puede ser en el entorno de la ingeniería aeroespacial. Esto

provoca que desde los años 70 el estudio de la fiabilidad de circuitos digitales haya ido aumentando. El estudio de la fiabilidad comienza con el conocimiento y control de los fallos que afectan a los circuitos digitales.

Los fallos en circuitos digitales se pueden clasificar en función de distintos aspectos como:

- La etapa de vida del producto en la que se produce. El fallo será de distinto tipo si se produce en la cadena de producción, durante las pruebas del producto o en servicio.
- El origen del fallo. Podemos separar dos fuentes de errores, las naturales y las artificiales. Siendo las primeras producidas por el entorno, como puede ser un ambiente corrosivo. Las artificiales hacen referencia a los fallos provocados por el ser humano.
- Naturaleza del fallo. La naturaleza del error cambia dependiendo de si afecta al hardware (un impacto físico) o al software (orígenes electromagnéticos).
- Intención. Podemos separar en dos tipos. Accidentales y deliberado.
- Duración del fallo. Dependiendo si el fallo es permanente, intermitente o transitorio.

Cabe destacar dentro de las distintas familias de tipos de fallo que acabamos de mencionar, los fallos producidos por causas naturales, en especial los producidos por la radiación cósmica. Estos fallos se conocen como Efectos de Evento Simple (SEE). Este tipo de fallo se empezó a detectar en satélites en la década de los 70 y con el tiempo ha ido aumentando debido a causas que comentaremos más adelante. Este fallo es producido de la siguiente manera:

La radiación cósmica está formada por partículas ionizantes. Estas partículas al atravesar un circuito electrónico crean una restructuración desigual de las cargas y huecos debido a la menor movilidad de estos últimos. Esto provoca que internamente se produzcan unas diferencias de potencial que generan corrientes puntuales no deseadas que pueden alterar el estado de biestables y celdas de memoria.

Este suceso puede provocar distintas consecuencias dependiendo de en qué parte del circuito se produzca. Nos centraremos en los errores que se producen en zonas de memoria del circuito digital. En este caso el error que produce un SEE es el cambio de un

bit del registro, lo que en el ámbito de investigación se conoce como "*bit flip*", es decir, en el registro afectado se cambia el estado de un bit.

En adición a la importancia que tiene el estudio de los SEE cabe destacar el aumento de este tipo de errores con el paso de los años. Esto es debido a la evolución de la electrónica, ya que cada vez se integra un mayor número de transistores en menor espacio, funcionando a niveles de alimentación menores por lo que la probabilidad de que una partícula ionizada genere una diferencia de potencial es mayor.

Dentro de los circuitos digitales, nos vamos a centrar en los sistemas empotrados en FPGA debido a las prestaciones que ofrecen y al creciente interés en utilizarlos en aplicaciones espaciales, donde los errores de tipo SEE son un problema importante.

Los sistemas empotrados consisten en un sistema con un microprocesador cuyo hardware y software están específicamente diseñados con el objetivo de que respondan a aplicaciones específicas, es decir, el diseño se orienta a una aplicación práctica en concreto con lo cual el diseño se ve más optimizado que si utilizáramos un PC de uso común para la misma tarea.

El sistema empotrado tiene distintas partes que podríamos resaltar:

- En primer lugar tenemos que destacar el microprocesador. Es la unidad central de proceso de información del sistema.
- Periféricos. Son subsistemas de entrada y salida, módulos de memoria, temporizadores, convertidores, o hardware de aplicación específica etc.
- Buses del sistema. Encargados de comunicar todo el sistema empotrado.

Como hemos mencionado anteriormente, el sistema empotrado consta de varias partes, por lo que los SEE podrían afectar a todas ellas.

Para estudiar la tolerancia a fallos de las distintas partes de un sistema empotrado en FPGA podemos realizar distintos experimentos como, radiación del sistema con partículas ionizadas, simulación de fallos, test de FPGA en funcionamiento o inyección de fallos a través del depurador del microprocesador.

El primero es el que realiza fallos más similares a la realidad, con una cantidad de fallos por segundo alta comparado con los producidos de forma natural por radiación cósmica, sin embargo es muy costoso debido al material utilizado en el experimento y solo se puede utilizar en sistemas finalizados, por lo que se usa para pruebas finales.

El segundo permite realizar fallos durante el proceso de diseño sin necesidad de instrumental extra pero los tiempos de simulación de fallos son demasiado elevados.

El tercero podría parecer la opción más lógica, ya que la prueba es el funcionamiento real del sistema. Este aspecto no solo aporta beneficios al método ya que la probabilidad de fallo con la que se diseñan los sistemas empotrados es muy reducida por lo que para presenciar fallos del tipo bit-flip necesitaríamos altos tiempos de experimentación.

En el caso concreto del microprocesador integrado en el sistema, se puede realizar el estudio de la tolerancia a fallos mediante una campaña de inyección de fallos a través del sistema de depuración que incluyen los microprocesadores actuales. Este método permite realizar la prueba de la fiabilidad en la etapa de diseño, mientras el sistema está funcionando y con una velocidad de fallos aceptable. En contra de este método podemos decir que tenemos registros que no son accesibles por depuración además de que el fallo inyectado no deja de ser un modelo lógico y no puede ser tan realista como otros modelos que someten el circuito a radiación directamente.

En este proyecto el trabajo se centra en estudiar la viabilidad de realizar una campaña de inyección de fallos mediante el depurador del microprocesador integrado en una FPGA, por la disponibilidad de los recursos necesarios y por la tasa de fallos inyectada.

1.1.1 Inyección de fallos en microprocesador a través de las estructuras de depuración

Como hemos mencionado anteriormente este tipo de inyección se enfoca a sistemas que están en desarrollo o en fase de pruebas, ya que no se precisa de más instrumentación que un PC para comunicarse con el depurador. Este método es el que nos permite representar más fielmente los fallos del tipo bit-flip que deseamos estudiar, es decir, los que se producen en zonas de memoria, ya que el efecto producido es el cambio de un solo bit de los registros o posiciones de memoria afectadas.

Una característica importante de este método es que nos permite realizar la inyección de fallos *on-line*, es decir, mientras el programa está funcionando.

Este proceso se realiza accediendo a los recursos internos del microprocesador a través del puerto de comunicación del depurador.

Para realizar este método necesitamos conocer las zonas accesibles de nuestro sistema, tanto del microprocesador como de periféricos. Una vez tenemos listadas las

direcciones accesibles por el depurador procedemos a realizar la inyección de fallos de la manera siguiente:

- En el código que va a ejecutar el microprocesador mientras se realiza la inyección, introducimos unos puntos de ruptura en el momento en que se desea inyectar el fallo
- Una vez tenemos el código con los puntos de ruptura arrancamos la campaña de inyección de fallos mediante el depurador. Cada vez que el código llegue a un punto de ruptura, a través del depurador tendremos acceso de lectura y escritura (esto es importante, ya que si no tuviéramos acceso de lectura no podríamos alterar en un solo bit los registros). De manera que mediante una operación matemática podemos alterar el valor leído del registro en un solo bit y escribirlo modificado, esta parte del proceso conviene tenerla programada ya que la finalidad del método es tener una alta velocidad de inyección, fin que no sería posible si tuviéramos que realizar estas modificaciones manualmente.
- Después de la inyección de fallos se continúa con la ejecución hasta que finaliza y se observa si la variación producida por estos repercute en el funcionamiento esperado.

En nuestro proyecto al ser un primer estudio de la viabilidad centraremos la inyección de fallos solo en el microprocesador empotrado dentro de una FPGA, con el fin de simplificar el sistema, sin afectar a los periféricos que forman el sistema. Nos comunicaremos con el microprocesador a través del sistema de depuración que este proporciona.

1.2 Objetivos

El objetivo principal de este proyecto es el estudio de la viabilidad de implementar un sistema software para la inyección de fallos. Este objetivo se realizará en nuestro sistema formado por un microprocesador de la familia ARM, en concreto el modelo Cortex-M1 empotrado en una FPGA del fabricante Actel. De manera que los objetivos a conseguir son:

- Conocimiento del sistema de depuración de los microprocesadores Cortex-M1. Conocer qué tipo de comunicación utiliza y cuál es su grado de accesibilidad en la arquitectura del microprocesador.
- Conocimiento de las herramientas para la creación de hardware y software de la FPGA así como la creación de tutoriales para el manejo de estas por nuevos usuarios.
- Puesta a punto de un sistema formado por la FPGA y un PC, que será el sistema sobre el que realicemos las pruebas experimentales necesarias.
- Realización de pruebas experimentales para ver el grado accesibilidad que podemos obtener en este tipo de sistema y conocer si es posible, y en caso afirmativo también saber cómo, realizar una campaña de inyección de fallos sobre el microprocesador Cortex-M1 empotrado en una FPGA.

1.3 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada uno de los capítulos posteriores:

- Estudio del microprocesador.

  En este capítulo procederemos a describir la estructura interna del procesador en todos aquellos aspectos que puedan sernos de utilidad para la inserción de fallos. De esta manera, prestaremos especial atención a las interfaces de comunicación del microprocesador y a las utilidades para la depuración.

- Estudio de la placa de la FPGA.

  Después de haber analizado el microprocesador internamente realizaremos una breve descripción de los elementos de los que dispone la placa de la FPGA.

- Instalación y uso de herramientas software.

  Se expone una breve guía en la que se encuentran todos los preparativos previos a la programación que hay que realizar en el sistema para que todo funcione correctamente así como la explicación detallada de las herramientas de las que dispone cada programa y de cómo utilizarlas con el fin de crear un programa que se ejecute en nuestra FPGA.

- Pruebas experimentales

  Se comenta los distintos tipos de experimentos prácticos que se han realizado y sus resultados.

- Conclusiones y trabajos futuros

  Se analizan los resultados obtenidos en las pruebas experimentales.

Capítulo 2

Estudio del Microprocesador

2.1 Características generales

La familia de microprocesadores en la que se centra el estudio es la familia ARM, más concretamente el modelo Cortex-M1. Es imprescindible hacer un análisis del microprocesador para saber cómo vamos a trabajar con la inyección de fallos. Habrá que conocer la estructura del microprocesador y más concretamente lo referente al sistema de depuración y su comunicación con el PC.

El microprocesador incorpora:

- Núcleo de procesamiento con las siguientes características:
    - Arquitectura ARM (ARMv6-M) de 32 bits.
    - Capacidad de ejecutar un SO.
    - Puntero de pila, dos en el caso de que el SO este activado.
    - Modelo para trabajar cuando el sistema entra en excepciones.
    - Siempre trabaja en el estado "Thumb". De manera que cuando esta activo el estado Thumb las instrucciones son de 16 bits y cuando esta activo el Thumb-2 las instrucciones son de 32 bits.
- Controlador de interrupciones NVIC (Nested Vectored Interrupt Controller). Las características incluyen:
    - El número de interrupciones externas a configurar: 1, 8, 16 o 32.
    - Selección mediante 2 bits del nivel de prioridad.
    - Guarda el estado del microprocesador a la entrada de una interrupción y lo devuelve cuando esta finaliza sin aumentar el número de instrucciones.
- Tanto memoria como interfaces AHB externas.
- Depuración completa o reducción de depuración que incluye:
    - Acceso de depuración a toda la memoria y registros del sistema incluyendo el banco de registros del micro cuando este se detiene.
    - Puerto de acceso para depuración
    - Implementación de *breakpoints* y *watchpoints*.

El microprocesador puede trabajar con depuración o sin ella, como es lógico, para nuestros objetivos, utilizaremos la configuración con depuración. En esta configuración los cinco bloques principales son:

- El Core.
- NVIC (Nested Vectored Interrupt Controller).

- El Bus Máster de comunicación.
- Bus periférico AHB-PPB.
- Depurador.

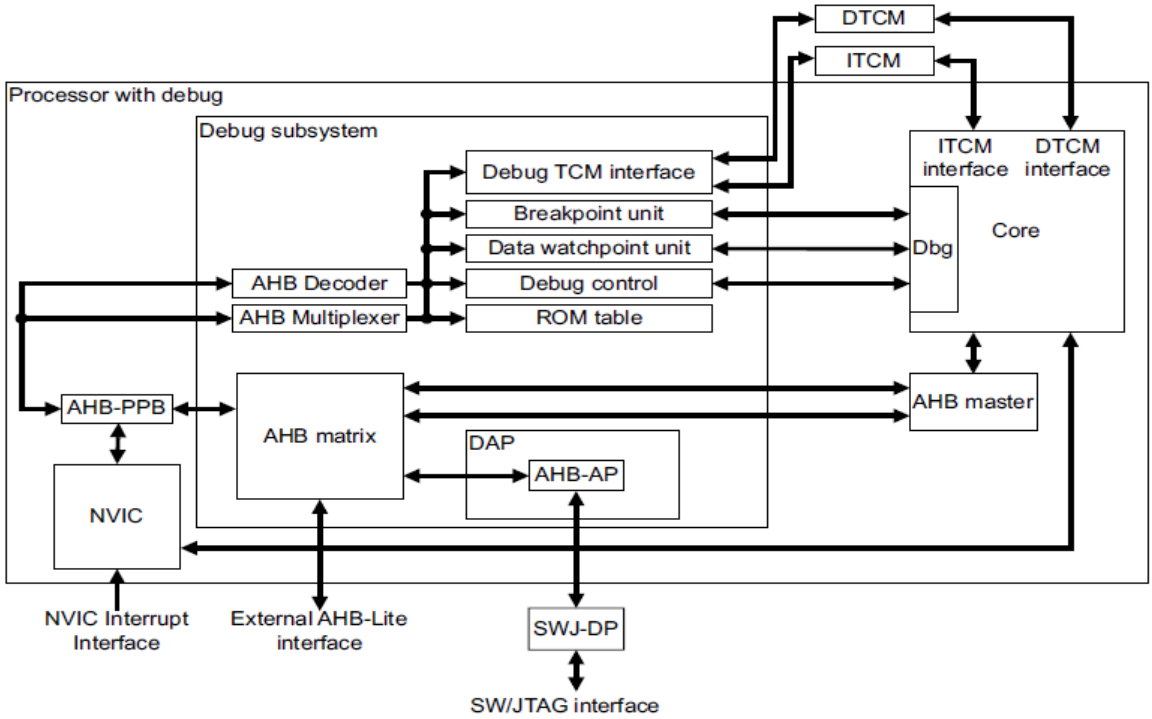


Figura1 Esquema del microprocesador

Observando la Figura1 y centrándonos en las partes del microprocesador que nos interesan a continuación analizaremos con más profundidad los estándares de comunicación AHB (Advanced High-preformance Bus), SW (Serial Wire) y JTAG (Joint Test Action Group), así como las posibles configuraciones de la depuración.

2.2 Registros de interés y mapa de memoria

Dentro del microprocesador hay una gran cantidad de registros accesibles total o parciamente. A continuación presentamos los que tienen más importancia de cara a la ejecución de un programa en el microprocesador:

- Registros de propósito general: esta familia de registros la forman los que podemos ver en la Figura2 a excepción de los registros PSR. Todos los registros

son accesibles por todas las instrucciones, menos los del R-8 al R-12, que no son accesibles por instrucciones de 16 bits, es decir cuando nos encontramos en el estado Thumb y no en el Thumb-2.

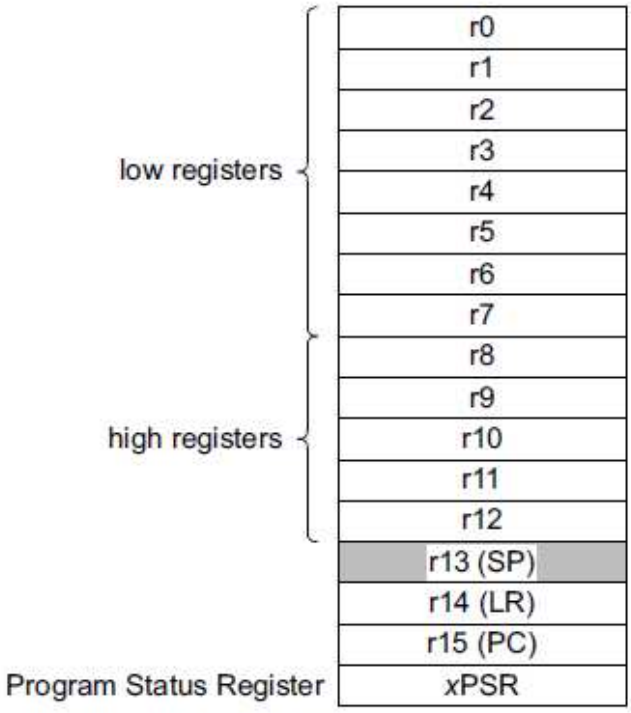


Figura2 Lista de registros generales

- El R13 es el puntero de pila o SP (Stack Pointer), el R14 el Link Register (LR) y el R15 el contador de programa o PC (Program Counter).
- Los registros PSR (Program Status Registers):
  - APSR (Application Program Status Register): contiene el código de condiciones de los flags.
  - IPSR (Interrupt Program Status Register): contiene el número de la actual activación de excepción.
  - EPSR (Execution Program Status Register): este bit se activa cuando el microprocesador sale del reset. Poner a cero este bit, mientras se está desarrollando el programa, provoca un fallo grave.

- En la Figura3 podemos observar el mapa de memoria del Cortex-M1. En este mapa podemos observar distintas zonas de interés:
    - La parte de memoria reservada al código se encuentra en la parte más baja del direccionamiento de la memoria (hasta la dirección 0x1FFFFFFF).
    - La memoria SRAM está dividida en dos bloques uno en el que se encuentra la memoria de datos y la memoria externa (es el bloque direccionado entre 0x20000000 y 0x3FFFFFFF) y otro bloque de memoria interna (direccionado entre las direcciones 0x60000000 y 0x9FFFFFFF)
    - Observamos cómo los periféricos están mapeados en memoria, lo cual podría servir de interés para trabajos futuros en los que la inyección fuera más completa.
    - Vemos una gran cantidad de direcciones de memoria que están reservadas por el fabricante, las cuales no son objeto de depuración.

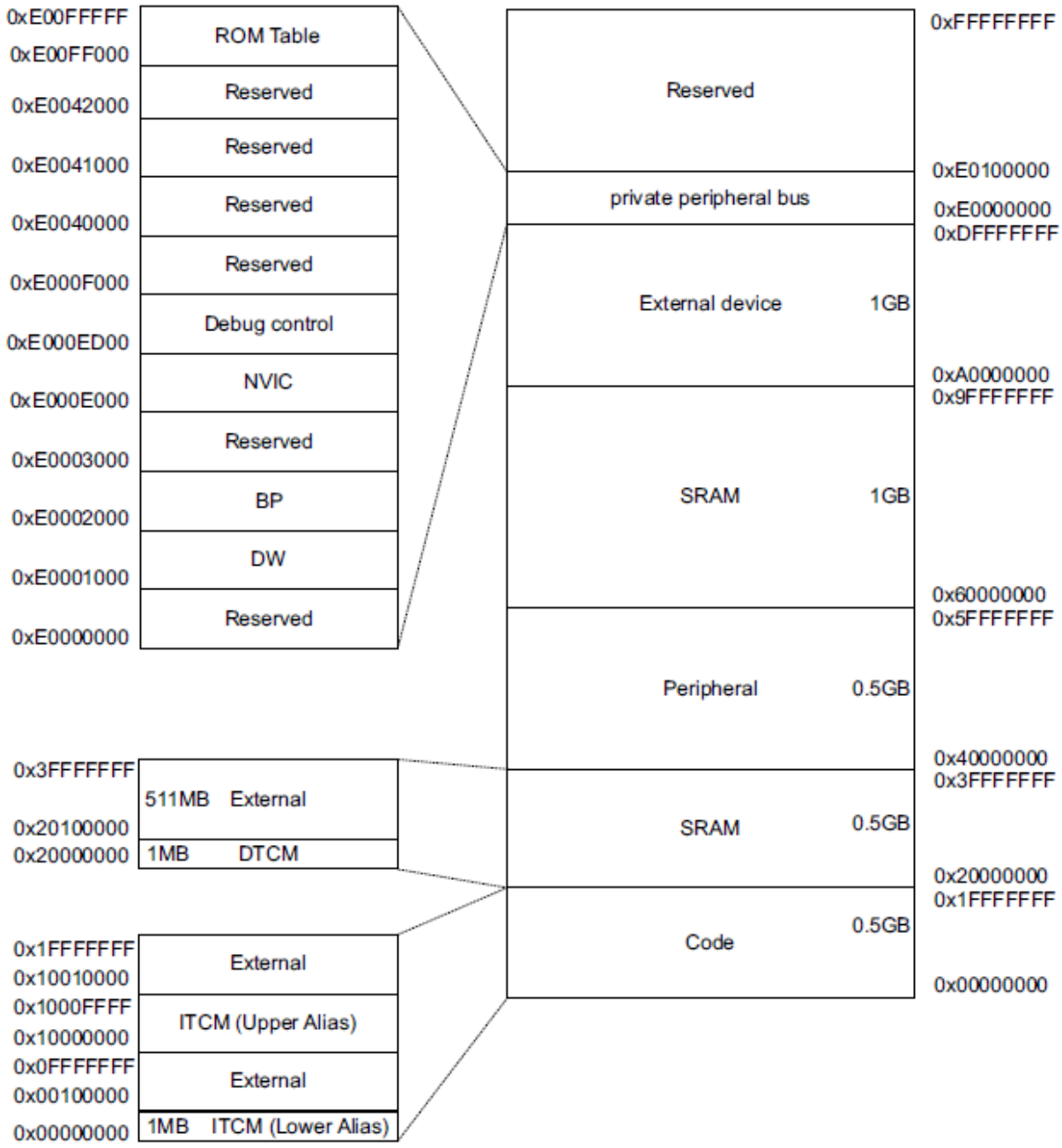


Figura3 Mapa de memoria

2.3 Comunicación en el Microprocesador

En nuestro microprocesador encontramos distintos protocolos de comunicación. Es necesario saber cómo funcionan estos para entender como fluye la información dentro del microprocesador. Como podemos apreciar en la Figura1 hay tres tipos de comunicación distinta:

- Estándar AHB. Utilizado para la comunicación interna del microprocesador.
- Estándar JTAG y comunicación en serie (RS-232). Son los modos de comunicación que se utilizan para comunicar el depurador con el exterior.

A continuación procedemos a explicar las características esenciales de funcionamiento en cada uno de ellos.

2.3.1 Estándar AHB

Las siglas vienen de las palabras *AdvancedHigh-performance Bus*, es decir, bus avanzado de altas prestaciones. Introducido por ARM a finales de los 90 cuenta con las siguientes características:

- Las instrucciones se realizan en los flancos de subida del reloj.
- Grandes anchos de banda (64/128 bits).
- Operaciones de comunicación divididas en cuatro fases. Direccionamiento, control, escritura y lectura. Como podemos observar en la Figura4, en cada ciclo de reloj solo se produce un direccionamiento, una asignación de control, una lectura y una escritura, sin embargo estas cuatro suceden simultáneamente para distintas comunicaciones.
- En cada fase de la comunicación solo puede haber un maestro, es decir, la comunicación esta segmentada en cuatro fases pero a cada fase solo se puede acceder de uno en uno. Esto hace referencia a lo mencionado anteriormente

sobre la Figura4, en la que solo se presenta una actividad de cada fase de comunicación por ciclo de reloj.

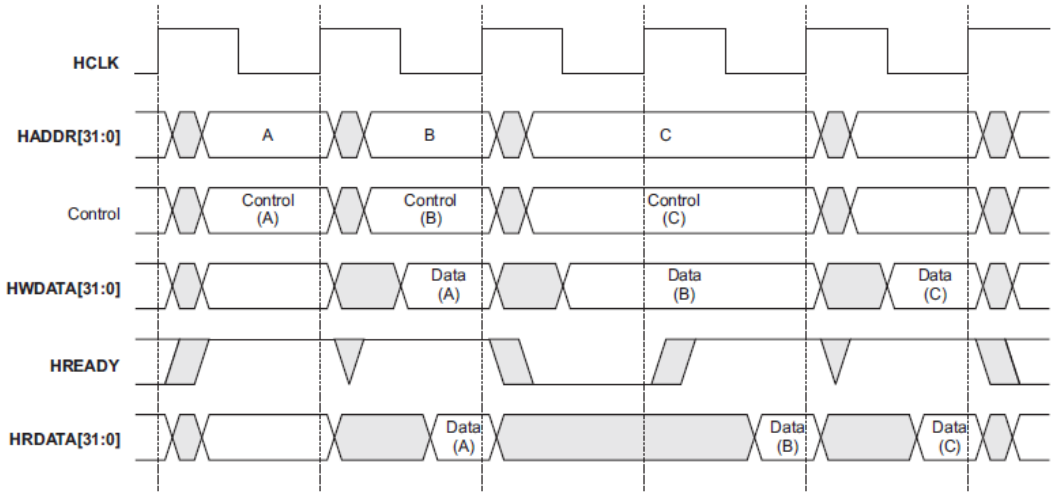


Figura4 Cronograma de comunicación AHB

Una transacción de datos del AHB consiste en una fase de direccionamiento (en la que se incluyen el direccionamiento y el control) y otra fase de transmisión de datos (en la que se incluye la lectura y escritura). El acceso al dispositivo de destino se gestiona mediante un multiplexor que no soporta el triestado, como hemos mencionado anteriormente, con lo que solo admite un maestro en cada bus por ciclo de reloj.

El AHB-Lite es un subconjunto del AHB que se creó en el siguiente estándar de ARM (2003). Este subconjunto solo permite un maestro en la comunicación, el core. De manera que elimina los algoritmos de asignación de recursos entre varios maestros.

2.3.2 Comunicación Serial Wire.

La comunicación SW, es decir, cable en serie, es la comunicación bit a bit a través de un canal. Antes solo se utilizaba para largas distancias, pero a pesar de que parece más lenta e inútil frente a la comunicación en paralelo, cada vez es mayor su uso en cortas distancias. Debido principalmente a la mejora en la integridad de la señal y a que las velocidades de transmisión en las nuevas tecnologías han empezado a superar la ventaja de la sencillez del bus en paralelo.

Otro posible motivo del uso del SW es el tema económico. En circuitos integrados el precio sube cuando aumentamos los pines de conexión, por lo que puede ser

interesante en aplicaciones donde la velocidad de comunicación no sea muy importante.

En concreto la arquitectura de SW utilizada por nuestra FPGA es la RS-232.

El RS-232 puede transmitir los datos en grupos de 5, 6, 7 u 8 bits, a unas velocidades determinadas (normalmente, 9600 bits por segundo o más). Después de la transmisión de los datos, le sigue un bit opcional de paridad (indica si el numero de bits transmitidos es par o impar, para detectar fallos), y después 1 o 2 bits de Stop. La configuración más utilizada es la 8N1, que significa: 8 bits de datos, sin paridad y con 1 bit de Stop.

El uso de este estándar en los años se ha visto reducido debido a la conexión USB, la cual necesita menores voltajes es más rápida y más fácil de instalar. En nuestro caso el medio físico de transmisión de datos es un USB con un driver para interactuar con el protocolo RS-232 de la FPGA.

2.3.3 Estándar JTAG.

Diseñado originalmente para ser usado en circuitos impresos, se utiliza en la prueba de submodulos de circuitos y también es muy común su uso como interfaz en la depuración de sistemas empotrados. Cuando se utiliza como interfaz de la herramienta de depuración, permite acceder al módulo de depuración que se encuentra integrado dentro del core. La interfaz JTAG utiliza cuatro o cinco pines, y está diseñada de tal manera que varios chips en una tarjeta puedan tener las señales JTAG conectadas en cadena, por lo que solo se necesita un puerto JTAG para acceder a todos los chips como se puede observar en la Figura5. Los pines son:

- TDI (entrada de datos de test).
- TDO (salida de datos de test).
- TCK (reloj de test).
- TMS (Selector de modo de test).
- TRST (reset de test) es opcional.

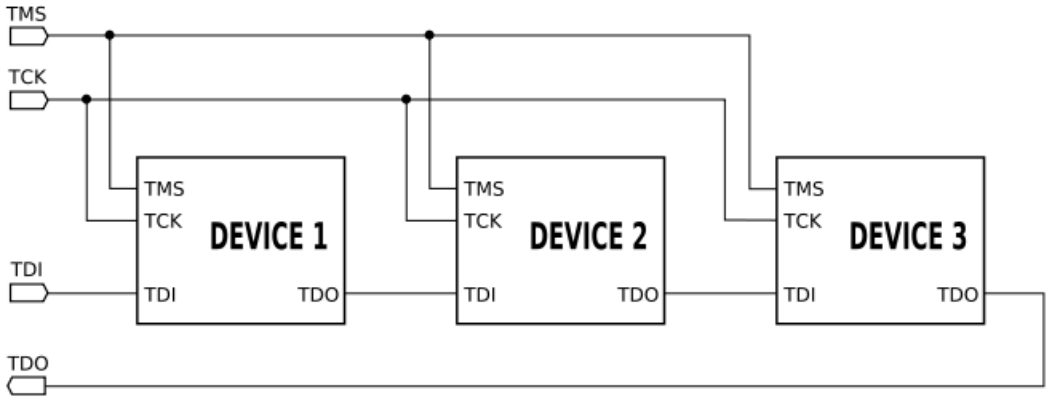


Figura5 Ejemplo de disposición para comunicación JTAG

Como se aprecia en la Figura1 el micro utiliza un estándar de comunicación interno, el AHB, y de cara a la comunicación externa tenemos una interfaz AHB y una interfaz SW/JTAG, la cual, está configurada como el puerto de acceso a depuración por lo que será la empleada para realizar nuestro trabajo.

La conversión entre estas dos señales se realiza en el DAP (Debug Acces Port), más concretamente en el puerto de acceso AHB, donde la señal que llega del puerto de depuración DP (Debug Port) es transformada a AHB-Lite.

El Cortex M1 soporta tres configuraciones seleccionables en las implementaciones del DP:

- Serial-Wire JTAG Debug Port: Combina el puerto de depuración del JTAG con el del SW (RS-232) e incluye un mecanismo que permite estar cambiando entre los dos.
- Solo Serial-Wire.
- Solo JTAG.

2.4 Análisis del depurador

Como hemos dicho anteriormente, el depurador es la vía para la inyección de fallos en el método que vamos a emplear. Por ello se incluyen unas nociones básicas acerca de la configuración, características, control y acceso al puerto de comunicación de este.

2.4.1 Configuración, características y control.

Principalmente el depurador tiene dos configuraciones posibles:

- La configuración completa posee cuatro breakpoints y dos watchpoints, es la configuración que viene por defecto.
- La configuración reducida posee dos breakpoints y un watchpoint.

El depurador permite realizar las siguientes funciones:

- Parada del core.
- Vuelta al funcionamiento normal o funcionamiento paso a paso.
- Acceso a los registros mientras la ejecución está detenida.
- Escritura/Lectura de:
    - TCM (Tightly-Coupled Memories). Es la interfaz que tiene la memoria del microprocesador para la depuración, tanto para datos (DTCM), como para instrucciones (ITCM).
    - Espacio de direccionamiento del AHB.
- Breakpoints. Permiten crear puntos de ruptura en el código que provocan que el programa se detenga.
- Watchpoints. Funcionan como los breakpoints pero permiten visualizar el valor de variables.

Las principales partes del depurador son:

- Los registros de control del depurador para acceder y controlar la depuración del core.
- BPU (unidad para implementar breakpoints).
- DW (unidad para implementar watchpoints).
- Interfaz de memoria del depurador para acceder a la TCM de datos e instrucciones.

- Tabla de memoria ROM. Muestra los registros mapeados en memoria ROM como las unidades de breakpoint y watchpoint o periféricos.

Todas las partes del depurador están internas en el PPB (Private Peripherial Bus) y como es lógico, su acceso solo es posible cuando la extensión de depuración está presente. Este acceso solo se puede realizar a través del puerto de depuración, el acceso a través de software está reservado.

El control de la depuración del microprocesador consta de una serie de registros accesibles que permiten al usuario realizar la configuración necesaria.

- El Debug Fault Status Register (DSFR) es el registro que controla el acceso al depurador, las solicitudes de parada, la ubicación de watchpoints y breakpoints y la captura de vectores.
- El Debug Halting Control and Status Register (DHCSR) informa del estado en el que se encuentra el microprocesador, habilita la depuración y permite el step y el stop del micro.
- El Debug Core Register Selector Register (DCRSR) sirve para seleccionar el registro del micro para recibir o trasmitir datos.
- El DebugCoreRegister Data Register (DCRDR) lleva a cabo la lectura de datos y/o la escritura de registros del core.
- El Debug Exception and Monitor Control Register (DEMCR) habilita la unidad de DW (data watchpoint) y causa la entrada en depuración de un determinado vector.

Unidad de Breakpoints:

Como hemos mencionado anteriormente dependiendo de la configuración que seleccionemos dispondremos de cuatro o dos breakpoints (cuatro por defecto).

Cada breakpoint se configura para realizar una parada en el hardware comparando la dirección en la que está situado con la dirección en la que se encuentra el código.

Para operar con esta unidad se utilizan dos registros:

- Breakpoint Control Register: utilizado para habilitar los breakpoints.
- Breakpoint Comparator Registers: que compara las direcciones de código con las de los breakpoints.

El microprocesador posee otra unidad para los watchpoints, pero debido a que no han sido de importancia para el proyecto, se omite su descripción.

2.4.2 Puerto de acceso a la depuración.

Cuando incluimos en la configuración del procesador al depurador, el procesador también incluye al AHB-AP (Acces Port) ya que es la interfaz interna de depuración del Cortex-M1.

Como ya hemos mencionado anteriormente este estándar se comunica externamente con el usuario a través del DP (puerto de depuración) mediante tres posibles opciones:

- SW.
- JTAG.
- La unión de ambos.

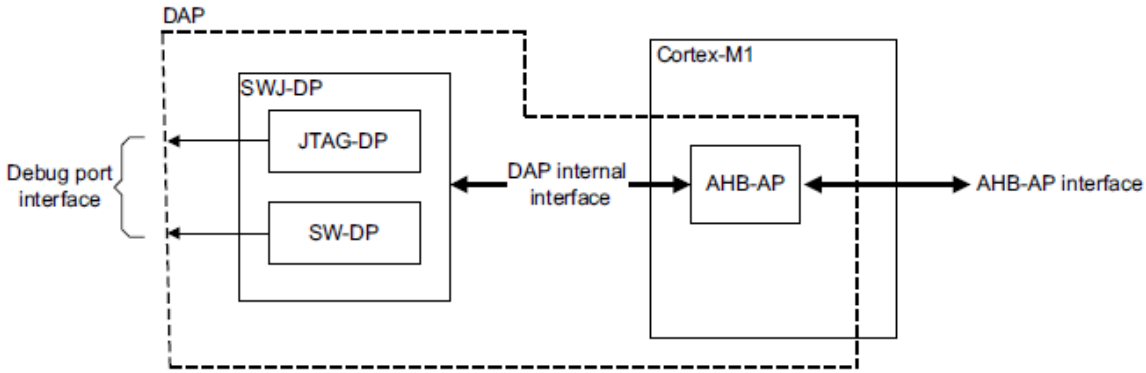


Figura6 Esquema del puerto de acceso para depuración

La Figura6 muestra la configuración del puerto de acceso a depuración, que como podemos observar lo forman el DP con el AP formando la zona que procedemos a llamar DAP (Debug Acces Port).

2.4.2.1 Acceso a depuración:

Para acceder a depuración se produce un cambio en el modo de transmisión de datos pasando del estándar AHB al estándar seleccionado para depurar (JTAG o RS-232).

El acceso a memoria con la DAP se hace de la misma manera que cuando el microprocesador funciona en régimen normal.

2.4.2.1.1 Acceso mientras se resetea el core:

Para que el acceso a través de depuración a todos los módulos sea posible en todo momento, hay que resetear toda la lógica de depuración llamando a una señal (DBGRESETn) diferente a la que resetea el sistema (SYSRESETn). La interfaz de depuración y lógica de acceso de depuración son accesibles cuando se utiliza SYSRESETn.

Las consecuencias de utilizar SYSRESETn son:

- Los cambios que realiza el depurador mediante escritura en los elementos que no forman parte de este, incluyendo los registros del core, no tienen efecto.
- Las lecturas que realiza el depurador a elementos que no forman parte del depurador, incluyendo los registros del core, devuelven datos imprevistos.
- Acceso completo a través de depuración al sistema de buses del AHB pero con datos de lectura imprevistos

2.4.2.1.2 Acceso mientras el core está ejecutandose:

Es importante saber que la lucha de prioridades entre el core y el depurador favorece siempre al depurador, esta es la razón por la que el DAP siempre tiene prioridad. Esto quiere decir que todos los accesos a través del DAP son posibles, pero puede cambiar los tiempos precisos de acceso del core.

2.4.2.1.3 Acceso del depurador al TCM:

Es importante tener en cuenta unas restricciones de advertencia:

- Utilizar el depurador para acceder a escritura de TCM solo cuando el core está parado.

- Aunque el depurador puede proporcionar acceso de depuración al TCM mientras el microprocesador está corriendo, algunas implementaciones de que estén almacenadas en RAM pueden dar resultados imprevistos cuando una lectura y escritura ocurren al mismo tiempo. Habrá que tener en cuenta esta restricción a la hora de implementar los fallos, de manera que el acceso sea jerárquico y no ocurran superposiciones de señales.

Capítulo 3

Estudio de la placa de pruebas

3.1 Herramienta Hardware

En este apartado procedemos a presentar la herramienta de hardware con la que se ha realizado el proyecto.

La herramienta hardware es una FPGA de Actel, como ya se ha mencionado anteriormente. Esta placa incluye los distintos dispositivos que pueden formar parte de nuestro sistema embebido, en particular el microprocesador Cortex-M1, que es el objeto de interés en este proyecto.

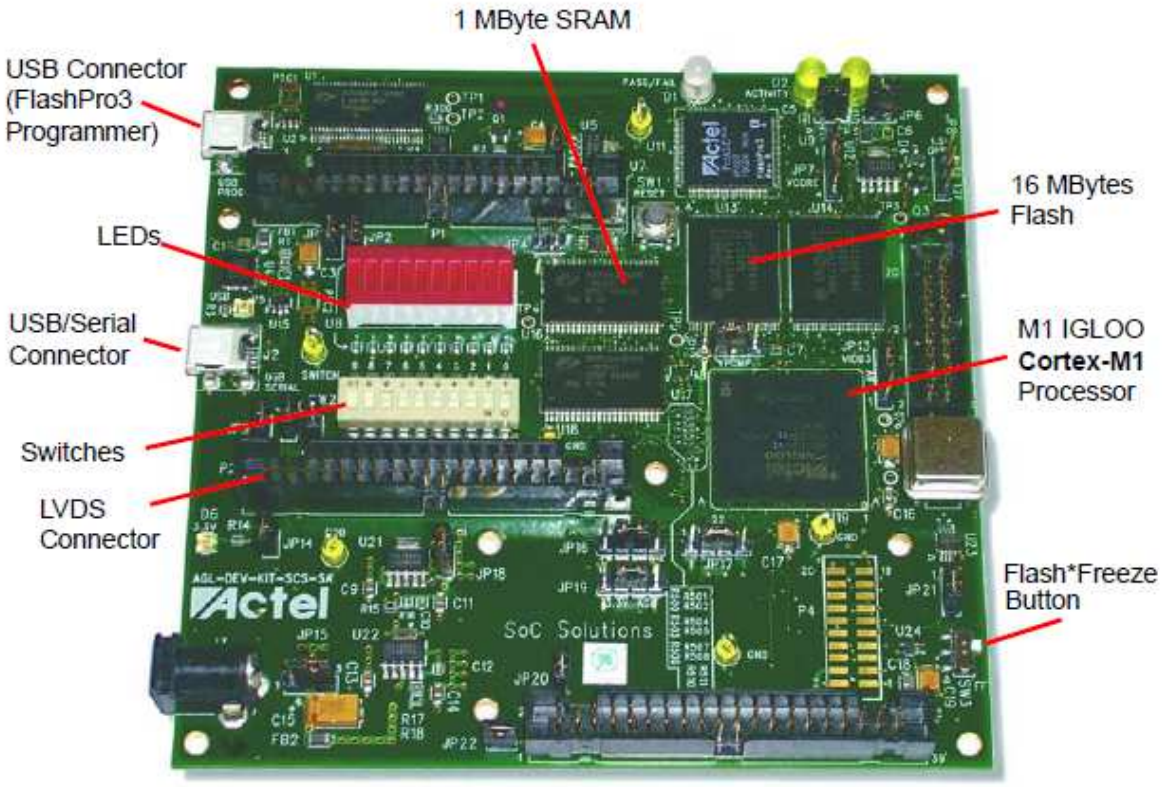


Figura7 Ilustración de la placa de pruebas

Las partes de la placa que más nos interesan para nuestro propósito son:

- LEDs de funcionamiento: como se describe más adelante los LEDs situados en la parte superior derecha de la Figura7 (dos amarillos y uno blanco) juegan un papel de comunicación con el usuario a la hora de verificar que la conexión FPGA-PC sea correcta así como las transmisiones de datos.
- El botón de reset: situado encima de la memoria SRAM nos facilita la reiniciación de la configuración del programa antes de realizar las pruebas. Es necesario pulsar este botón antes de realizar una nueva configuración de la FPGA, puesto que si no podría producirse un error o incluso no llegar a finalizarse la programación. Este error se ha observado experimentalmente en varias ocasiones en el desarrollo del proyecto.
- Las conexiones USB, a través de las cuales se realizarán las conexiones con el PC tanto para transferir programas como para realizar la depuración.

Adamas los principales elementos y características de la placa son:

- Tecnología Flash reprogramable.
- Niveles de voltaje a 1,2 ó 1,5 voltios.
- Alta capacidad, pines entrada/salida.
- 1 MByte SRAM.
- 16 MByte flash.
- Chip convertidor USB–RS232.
- Conectores GPIO.
- 20 pines conexión JTAG para el Cortex M-1.
- Oscilador de cristal.
- Conectores de expansión de memoria.

Esta serie de características y componentes pueden apreciarse mejor en la Figura8, la cual es un esquemático global de la placa:

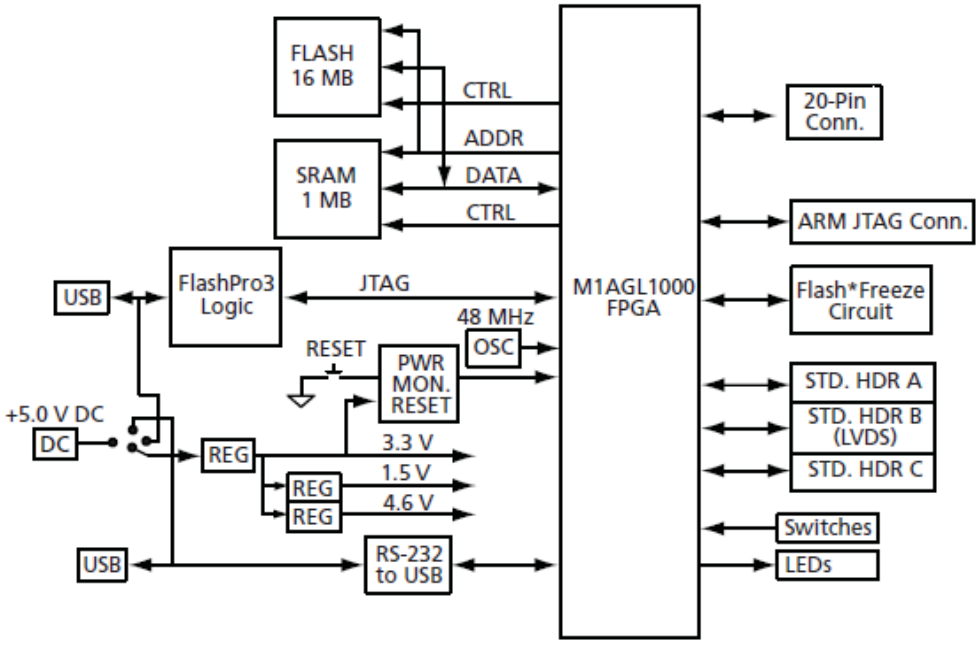


Figura8 Esquema de conexiones de la placa

Detalles del hardware a tener en cuenta:

La conexión de alimentación se realiza a través del conector J3, situado en la zona inferior izquierda de la Figura7. Para que la alimentación se pueda llevar a cabo hay q tener en cuenta la posición del jumper 22 situado al lado del J3. En (4) encontramos que la posición de Jumper debe ser la 3-5, pero la placa funciona con la posición 1-4 (además solo hay 4 pines por lo que debe ser una errata del fabricante).

Existen dos conexiones USB para comunicar con el PC. La J1, situada arriba a la izquierda en la Figura7, sirve para descargar los programas y arquitecturas. La J2, situada en el medio a la izquierda en la Figura7, sirve para la comunicación de depuración.

Ambas comunicaciones se pueden realizar simultáneamente y es posible su desconexión después de que la placa se haya programado, para nuestro caso, dado que queremos mantener una comunicación para la depuración dejaremos el USB asignado a esta tarea conectado.

Capítulo 4

Instalación y uso de herramientas Software

4.1 Instalación de herramientas Software

4.1.1 Descarga

Actel dispone de su propia herramienta para trabajar con la FPGA. La podemos conseguir directamente de la página de Actel, realizando los pasos descritos a continuación:

- Dirigirse a la dirección (1)
- Seleccionar el Libero SoC v10.1

- Seleccionar la aplicación para el sistema operativo del usuario (en este caso se ha realizado para Windows 7). Hacer click en la pestaña download.

A continuación saldrá una ventana para registrarse en la página, es importante recordar el usuario y contraseña así como el correo, ya que la utilizaremos más adelante para activar la licencia del producto.

Una vez nos hemos registrado e introducido nuestros datos aparecerá una ventana con las diferentes fuentes de descarga, tal y como se ve en la Figura9:

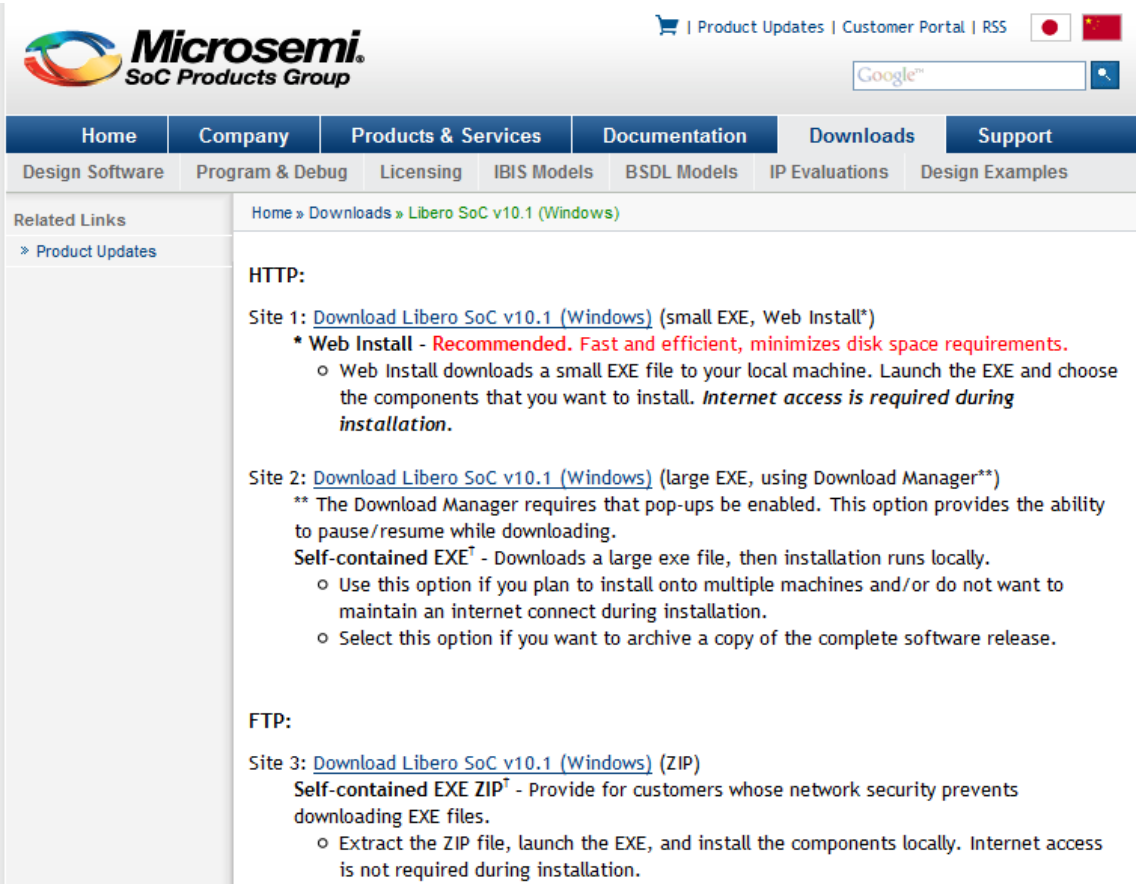


Figura9 Fuentes de descarga del Libero SoC

Como se ve en la Figura9 hay 3 posibles descargas. La primera opción realiza una descarga de un ejecutable pequeño que arranca una instalación online. Aunque parezca más cómoda esta opción, no la recomendamos, ya que se han encontrado problemas a la hora de reinstalar el programa o hacer variaciones, ya que al eliminar el programa instalado de esta manera hay subcarpetas de aplicaciones que no se eliminan bien y al reinstalar el programa dan problemas.

Sin embargo la segunda opción descarga un paquete de instalación completo que evita este tipo de problemas a la hora de configurar la instalación o la desinstalación.

A la hora de elegir la ubicación de instalación es conveniente elegir una dirección que no tenga espacios, ya que esto da problemas a la hora de crear proyectos en las aplicaciones. Por lo tanto es preferible instalarlo directamente en las unidades de disco y sobre todo evitar direcciones como *Mis Documentos*.

4.1.2 Validación de la licencia del producto

Al finalizar la instalación del programa nos preguntará si poseemos la licencia del producto. Si no se tiene, es necesario obtenerla ya que el programa no se ejecuta sin esta. Esta licencia es gratuita y se obtiene en la propia página de Actel.

Como se aprecia en la Figura10 en la parte superior de la página, en la pestaña de *Downloads*, hacemos click en la pestaña *Licensing*.

Figura10 Pestañas superiores de la página de Actel

Hacer click en el botón *Request Free License,* tal como muestra la Figura11.

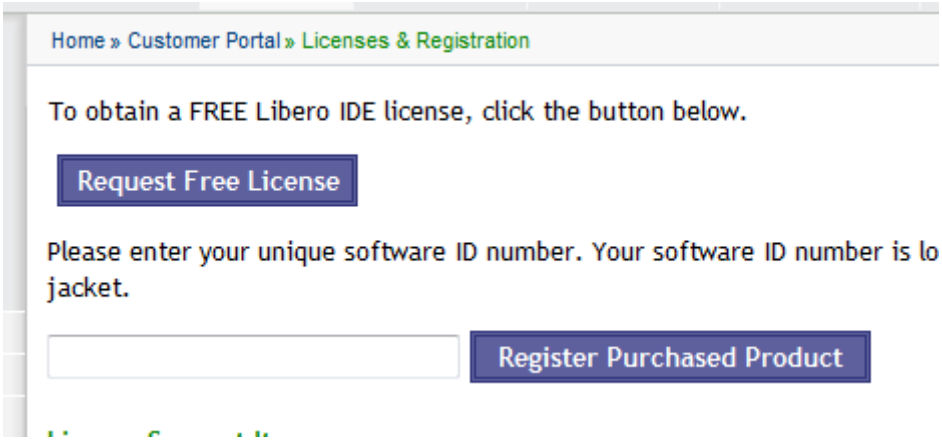


Figura11 Instrucciones de obtención de la licencia

Seleccionamos la licencia Gold para Windows (esta licencia es para un año).

A continuación Actel solicita la identificación del disco duro del PC, con el fin de no tener duplicados de licencias y llevar un control.

Este identificador lo podemos obtener en la consola CMD introduciendo el comando:

C:>Vol C:

Introducimos los dos conjuntos de 4 caracteres (incluyendo el guión). En un periodo de 30 minutos recibiremos un correo de Actel en el que nos explica como activar la licencia. Él correo viene con la licencia adjunta en un archivo con el nombre de *License.dat*.

4.1.3 Conexión de los puertos de comunicación

Antes de analizar las herramientas conviene instalar correctamente los drivers que se necesitan para descargar los programas en la placa así como para mantener la comunicación con esta. Esta instalación puede ser más complicada, de lo imaginable en un principio debido a que en el pack de instalación no se encuentran los drivers de comunicación.

En la página de SiliconLabs podemos encontrar un driver bastante completo para diferentes conversiones entre conexiones USB, la página se muestra en la Figura12, la dirección es (2):

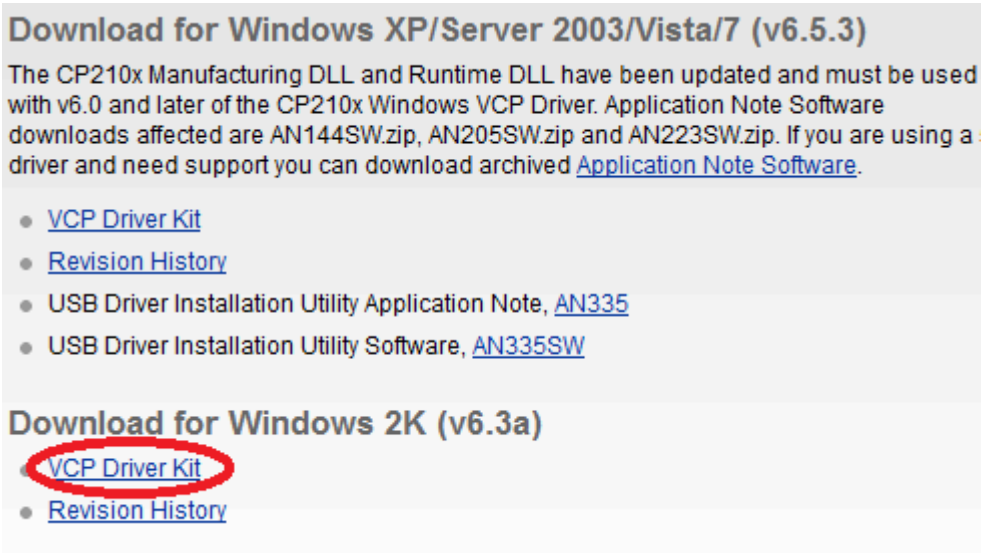


Figura12 Pagina de descarga de drivers de comunicación

Una vez instalado el driver procedemos con la conexión de la placa:

- Primero alimentamos la placa en solitario (sin conectar los USB).

- A continuación conectamos el USB de la conexión J1.

  Esta conexión no suele dar problemas. Para garantizar que la conexión se ha realizado con éxito se encenderá un LED amarillo situado en la parte superior derecha de la Imagen 5. Si queremos estar más seguros podemos abrir el administrador de dispositivos (Figura13) del panel de control y ver que la conexión utilizando el driver FlashPro3 se ha realizado:

Figura13 Comprobación comunicación del J1

  Como se observa la conexión se ha configurado.

- Después conectamos el USB a la conexión J2.

  La conexión se comprobaría de la misma manera que con la conexión anterior, accediendo al administrador de dispositivos y en este caso mirando en los puertos COM además de comprobar que el LED verde próximo a la conexión J2 se encienda. Esta conexión suele dar más problemas debido a que no se encuentran los drivers automáticamente, para solucionar el problema tenemos que instalar los drivers manualmente.

4.2 Tutorial para la herramienta Libero SoC

Una vez tenemos nuestro periférico con los puertos de comunicación correctamente instalados, procedemos a familiarizarnos con la primera herramienta Software, el Libero Soc. Esta herramienta se encarga de configurar la arquitectura hardware de la FPGA. Como ya hemos descrito anteriormente vamos a realizar un diseño para un Cortex M1.

A continuación procedemos a explicar cómo empezar un diseño desde cero, aunque para el proyecto usaremos una base de diseño proporcionada por Actel en un ejemplo.

4.2.1 Creación de un nuevo proyecto y diseño de bloques

Lo primero es aprender a crear un proyecto nuevo. Arriba a la derecha tenemos la pestaña project, la cual podemos desplegar, accediendo a la opción New Project, como muestra la Figura14.

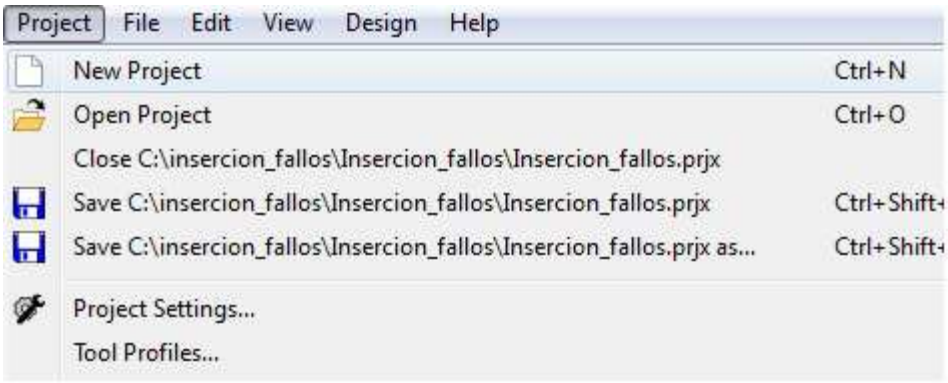


Figura14 Creación de un nuevo proyecto en Libero Soc

Se nos despliega una ventana como la que muestra la Figura15, donde rellenaremos los huecos del nombre del proyecto, su ubicación y la familia de la placa con la que estamos trabajando.

Las opciones de velocidad, voltaje y condiciones de operación, se rellenan automáticamente cuando introducimos la opción del encapsulado (Package).

Es importante que la ubicación del proyecto no tenga espacios, como ya hemos mencionado anteriormente, ya que dará problemas de compilación.

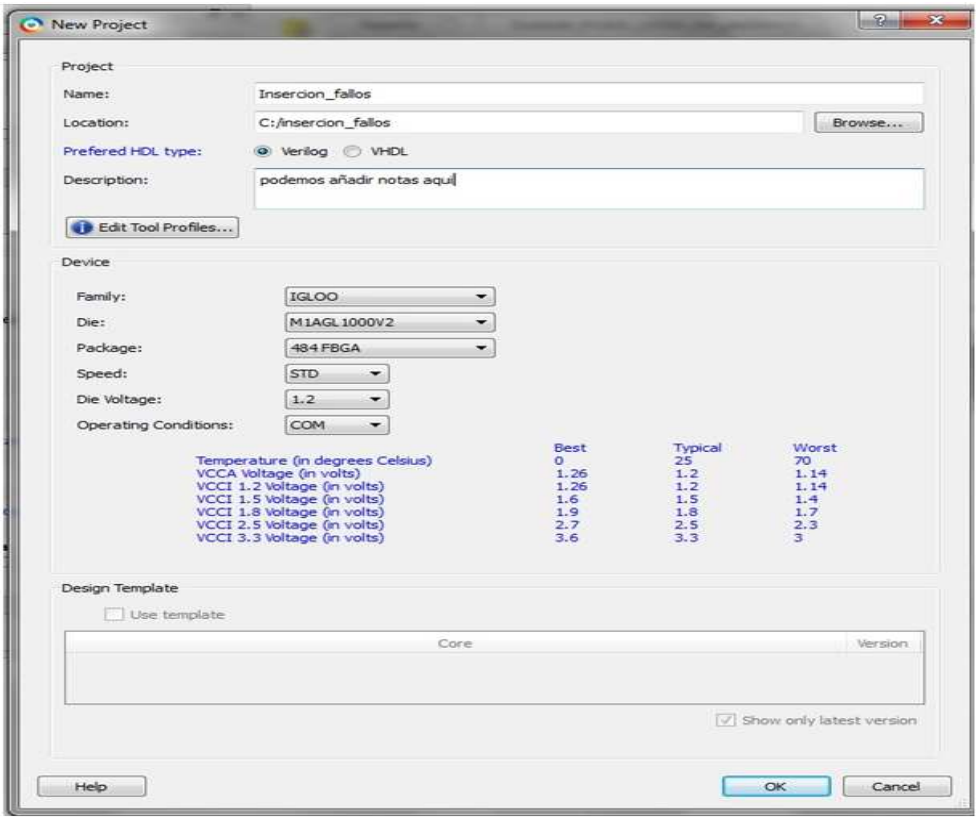


Figura15 Propiedades nuevo proyecto en Libero SoC

Una vez creado el proyecto vamos a ver cómo crear el diseño de los bloques de nuestro sistema.

En la parte izquierda de la interfaz disponemos de distintas subventanas (podemos agregar más), una de ellas es la de *designFlow* (Figura16), que es en la que nos centraremos para seguir paso a paso el diseño hardware.

Para crear el diseño de bloques (llamado SmartDesign en el programa), hacemos doble click en la tarea *CreateSmartDesign*.

Lo primero que tendremos que hacer es introducir un nombre para el diseño, si el nombre no cumple las restricciones del lenguaje HDL, seremos avisados y tendremos que elegir otro nombre distinto.

Una vez hemos introducido un nombre apropiado se genera la ventana para el nuevo diseño. Esta ventana dispone de todas las herramientas que se pueden utilizar para el diseño del sistema, colocadas en la parte izquierda.

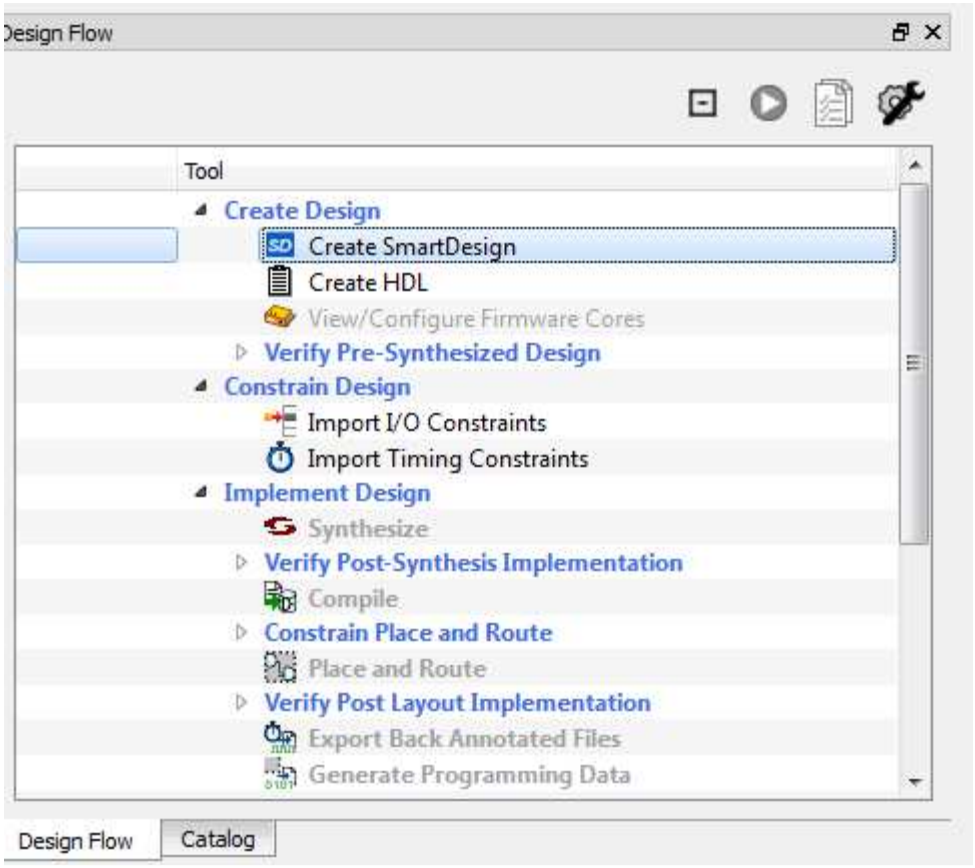


Figura16 Menú DesignFlow de Libero SoC

Es recomendable utilizar una herramienta que proporciona el Libero Soc en la parte superior de la interfaz (Figura17), que permite expandir las subventanas que estén situadas en la parte derecha de la interfaz, como pueda ser por ejemplo el diagrama de bloques del *SmartDesign.*

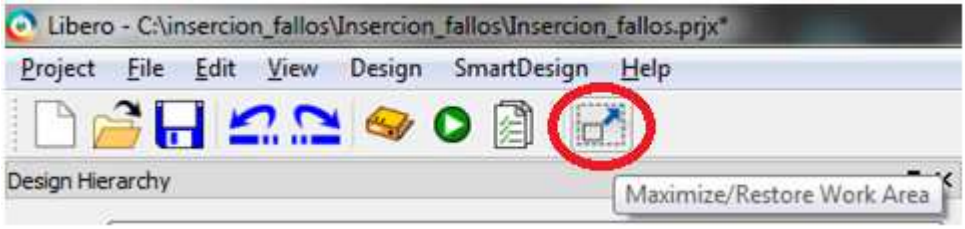


Figura17 Barra de herramientas superior de Libero SoC

Para volver al tamaño original y así poder ver el resto de herramientas volvemos a hacer click en la misma herramienta.

Para poder introducir módulos en el diseño tenemos que importarlos del catalogo (Figura18). Este se encuentra ubicado con el *DesignFlow* en otra pestaña:

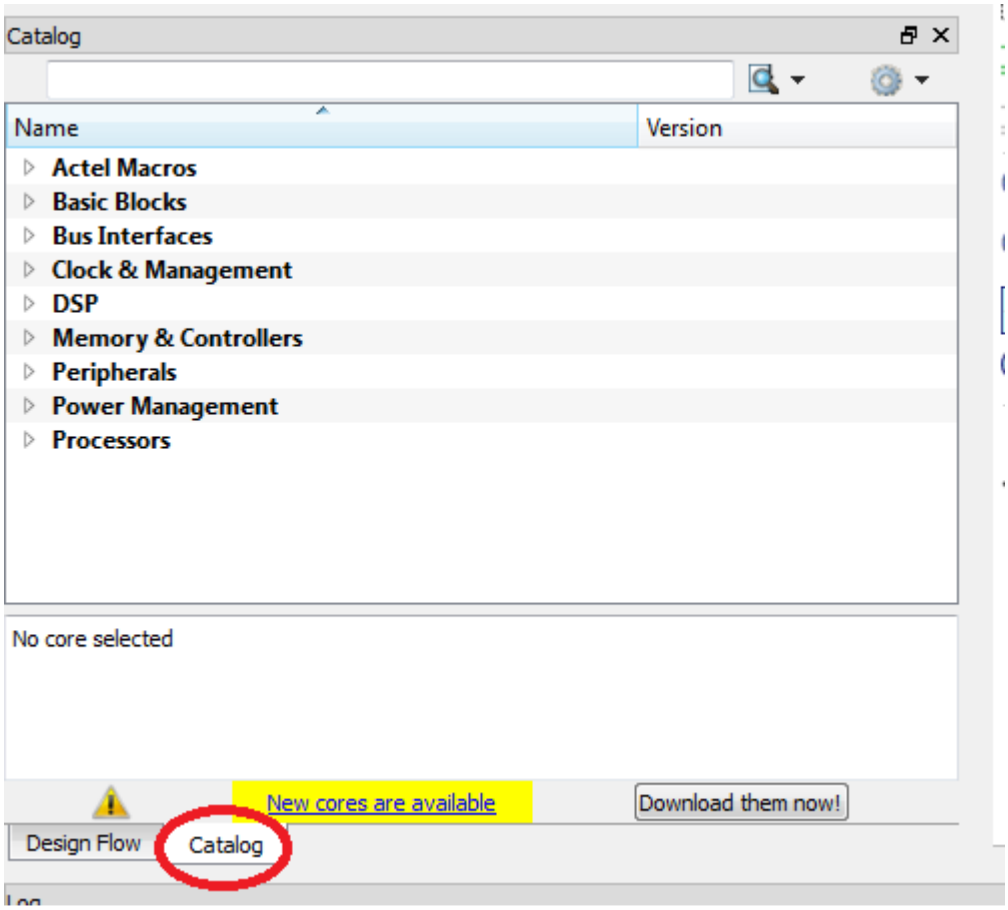


Figura18 Catalogo del Libero SoC

Es recomendable descargar los cores que haya disponibles asi como ultimas versiones que haya proporcionado Actel, con el fin de poder evitar posibles fallos que tuviesen las primeras versiones ademas de obtener perifericos que se hayan añadido posteriormente.

Procedemos a buscar el que nos interesa en la parte superior de la ventana, como indica la Figura19:

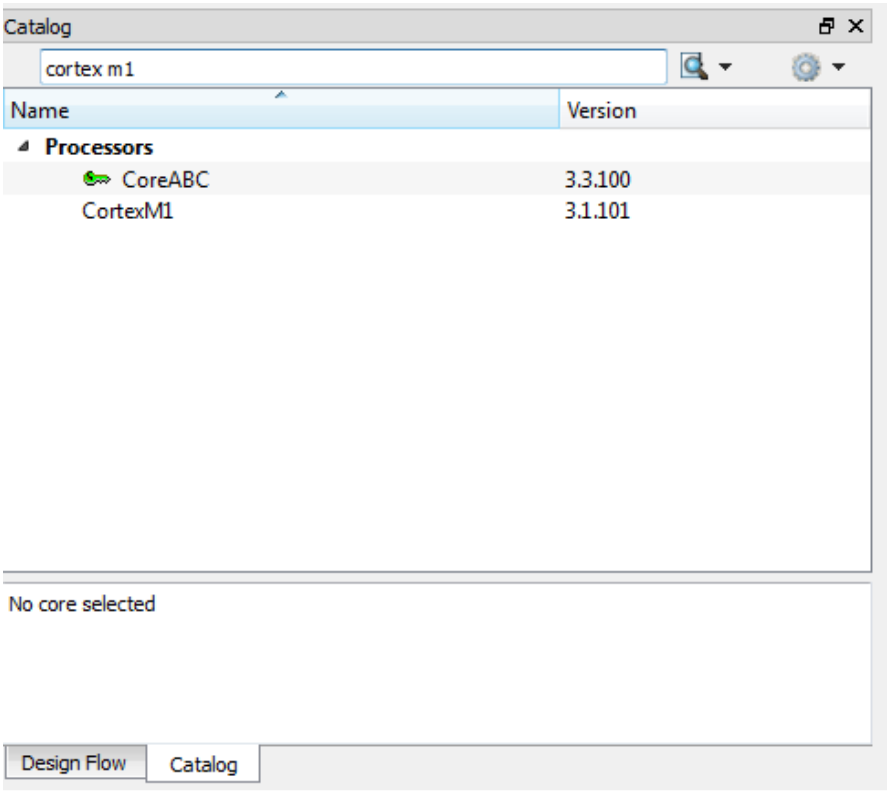


Figura19 Búsqueda en catálogo del Libero SoC

Al elegir, por ejemplo, el Cortex M1, aparece una ventana con la configuración inicial del Cortex (Figura20):

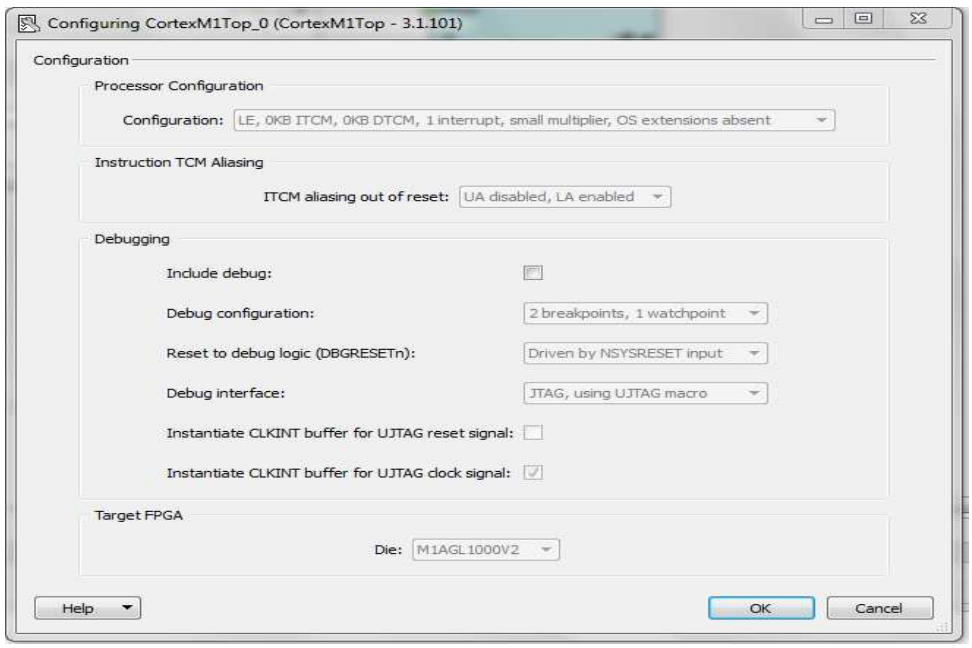


Figura20 Ventana de presentación de los módulos agregados

Al darle a OK se introduce el elemento en el diseño (Figura21), el cual forma un bloque con sus entradas y salidas. Para reconfigurar algún aspecto basta con hacer click con el botón derecho y acceder a configuración.

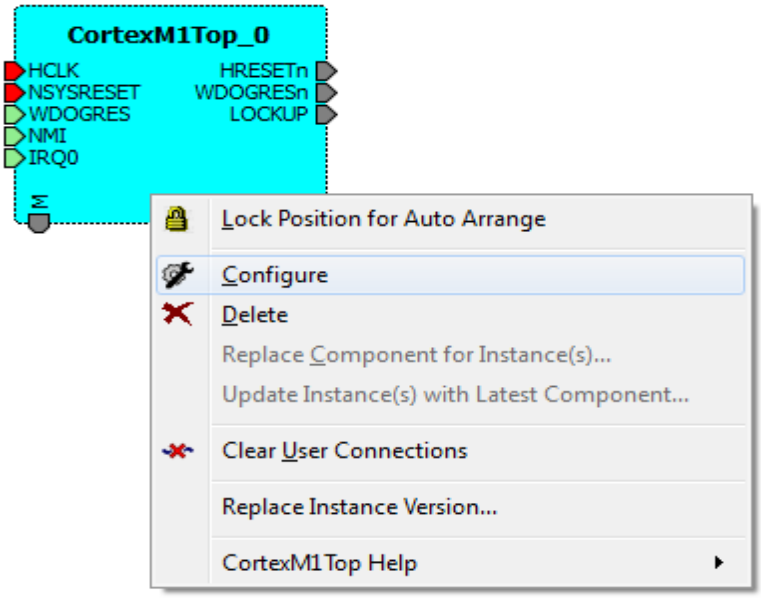


Figura21 Bloque insertado en SmartDesign

De esta manera podemos introducir todos los elementos que necesitemos, ya sean buses de comunicación o periféricos de cualquier tipo. Por ejemplo en la Figura22 se observan algunos posibles módulos.

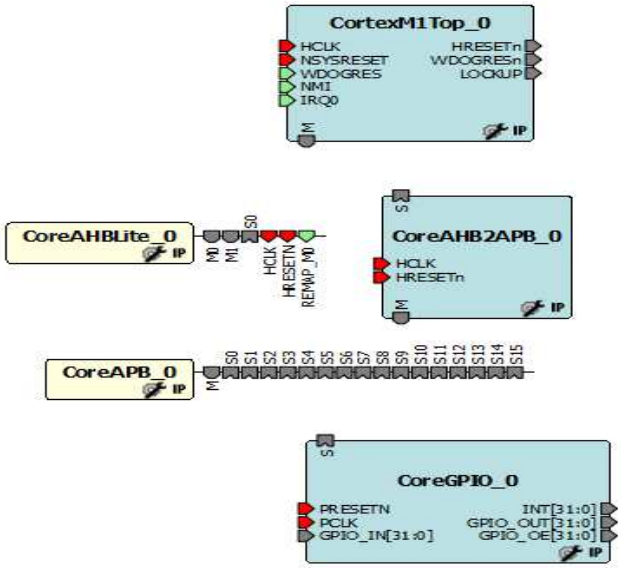


Figura22 Ejemplo de distintos módulos en SmartDesign

A continuación vamos explicar las distintas herramientas de las que consta el entorno del Smartdesign, las cuales se pueden apreciar en la Figura23, para poder realizar nuestro diseño:

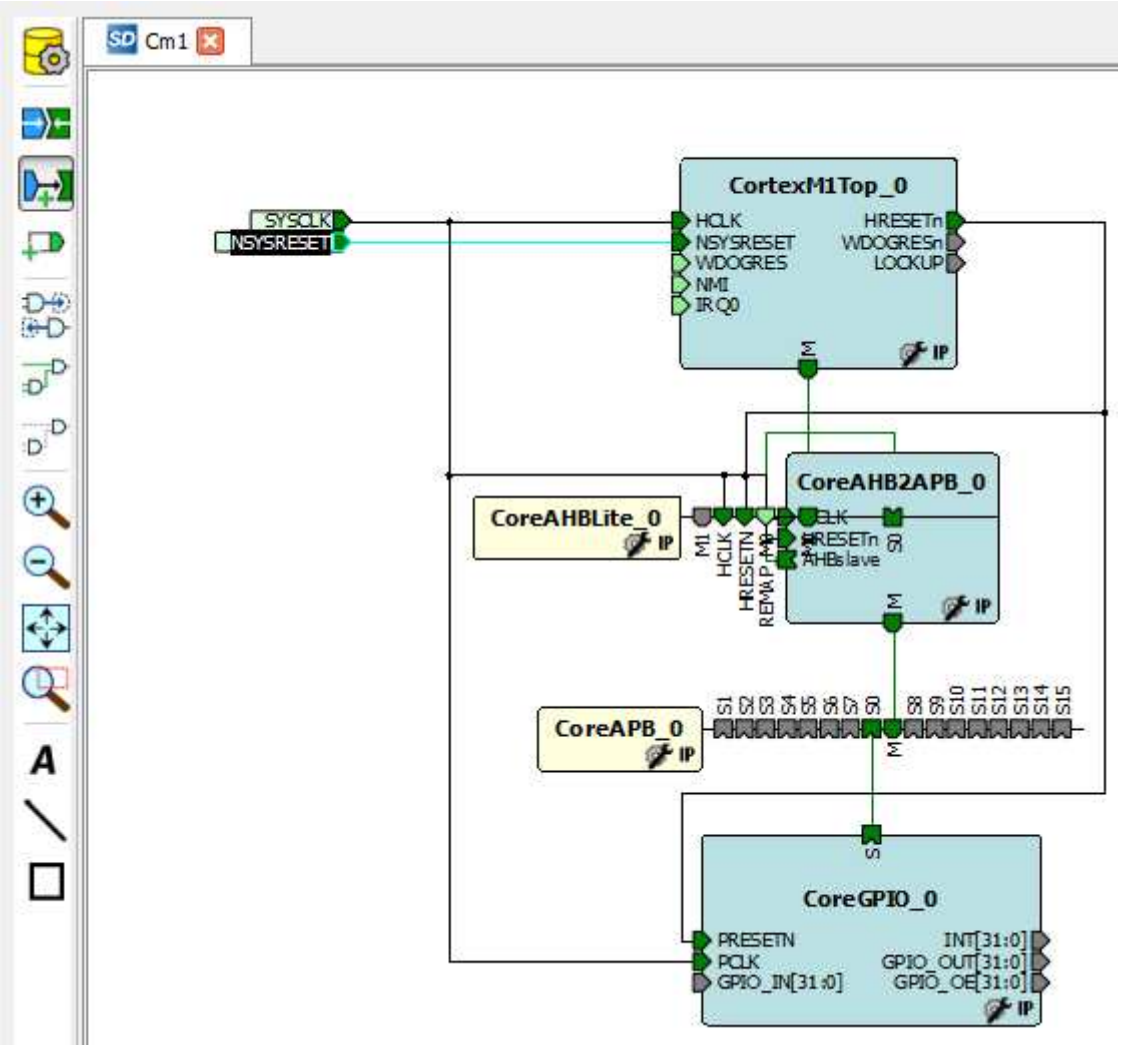


Figura23 Entorno de diseño del SmartDesign

- La primera herramienta (el cilindro amarillo) es la herramienta de compilación, es la que se encarga de encapsular el diseño, avisándonos en el caso de que nuestro diseño no cumpla todos los requisitos, por ejemplo, algún error en la conexión de los componentes.
- La segunda herramienta se llama *Auto Connect* y es de gran utilidad, al utilizarla el programa realiza con buena lógica la conexión entre los distintos bloques

buses y señales del diseño. En la Figura23 las conexiones se han realizado de esta manera.

- Si la anterior herramienta no realiza las conexiones de manera correcta podemos realizarlas manualmente con la tercera herramienta. Simplemente, seleccionando la herramienta, hacemos click en el puerto del primer bloque que forma la comunicación que queremos crear y sin soltar arrastramos el cursor hasta el puerto de conexión del bloque que nos interese.
- La cuarta herramienta se encarga de añadir señales, como se puede apreciar en la Figura24, podemos elegir si queremos que sea de entrada de salida o ambas.
- Además con el nombre que le pongamos estaremos haciendo referencia al ancho del bus del puerto, es decir, en el caso de la Figura24, el puerto que se va a insertar tiene un ancho de 32 bits.

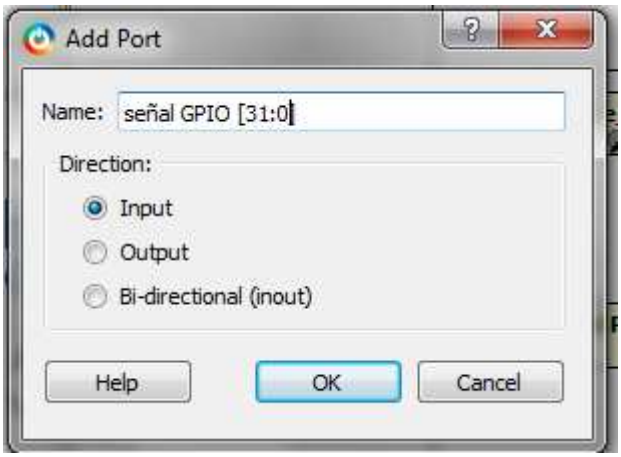


Figura24 Configuración de puerto en SmartDesign

- Las siguientes tres opciones sirven para la mejora de la visualización del diseño, ordenando la red formada por conexiones y bloques así como ocultando las redes si el usuario lo necesita.
- Las últimas opciones se encargan de la visualización del entorno, pudiendo alejar o acercar la vista, encuadrar el diseño o hacer zoom en una zona seleccionada. además disponemos de herramientas para introducir textos líneas o recuadros.

Con estas herramientas disponemos de toda la información para realizar el diseño de bloques (SmartDesign). El diseño estará finalizado cuando la compilación con la primera herramienta no de errores.

Podemos observar el diseño terminado de Figura25, el cual es el que utilizaremos como base del hardware en nuestras pruebas experimentales.

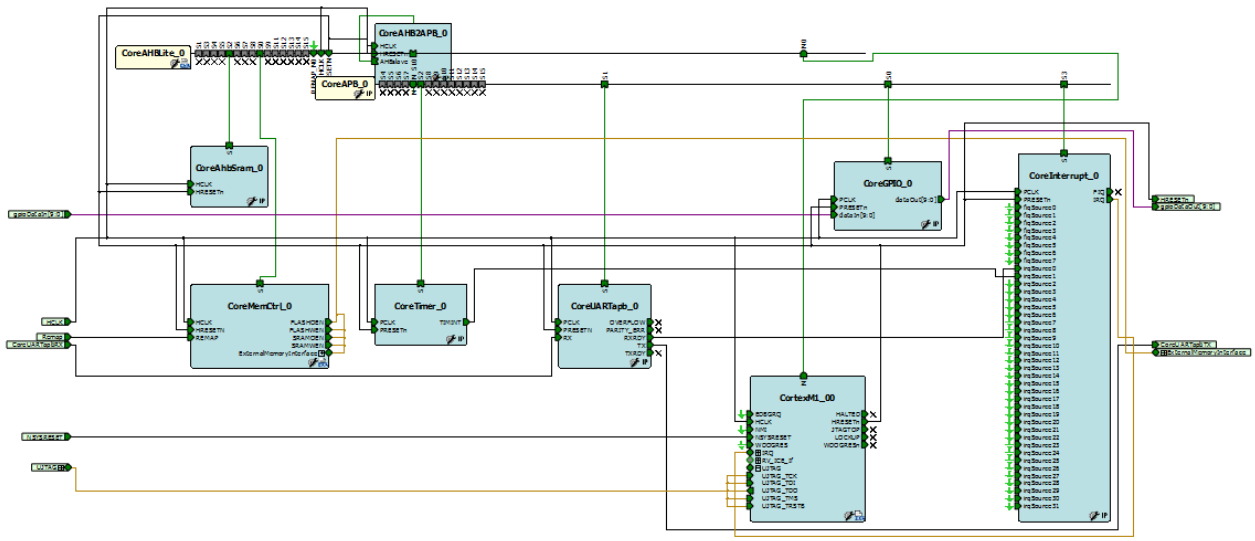


Figura25 Diseño Hardware completo

El siguiente paso a realizar es la adición de restricciones en las entradas y salidas o en la frecuencia del sistema, de manera que los puertos de entrada y salida que hemos creado en el SmartDesign se asocien con entradas y salidas físicas de la placa y la comunicación se realice a la velocidad deseada.

La edición en este punto se realiza a través de la inserción de archivos .sdc para la configuración de frecuencia del sistema y .pdc para la de conexión de entradas y salidas con los pines de la FPGA.

En este estudio no se ha profundizado en la configuración de tiempos. Si se desea más información acerca de la frecuencia del sistema, consultar (3).

4.2.2 Configuración y programación del diseño en FPGA

Cuando ya tenemos compilado nuestro SmartDesign es hora de realizar los pasos necesarios para traspasar nuestro diseño en el PC a la placa.

La herramienta Libero SoC presenta en el "DesignFlow" los distintos pasos que vamos a tener que ejecutar para validar distintas restricciones tal y como podemos ver en la Figura26.

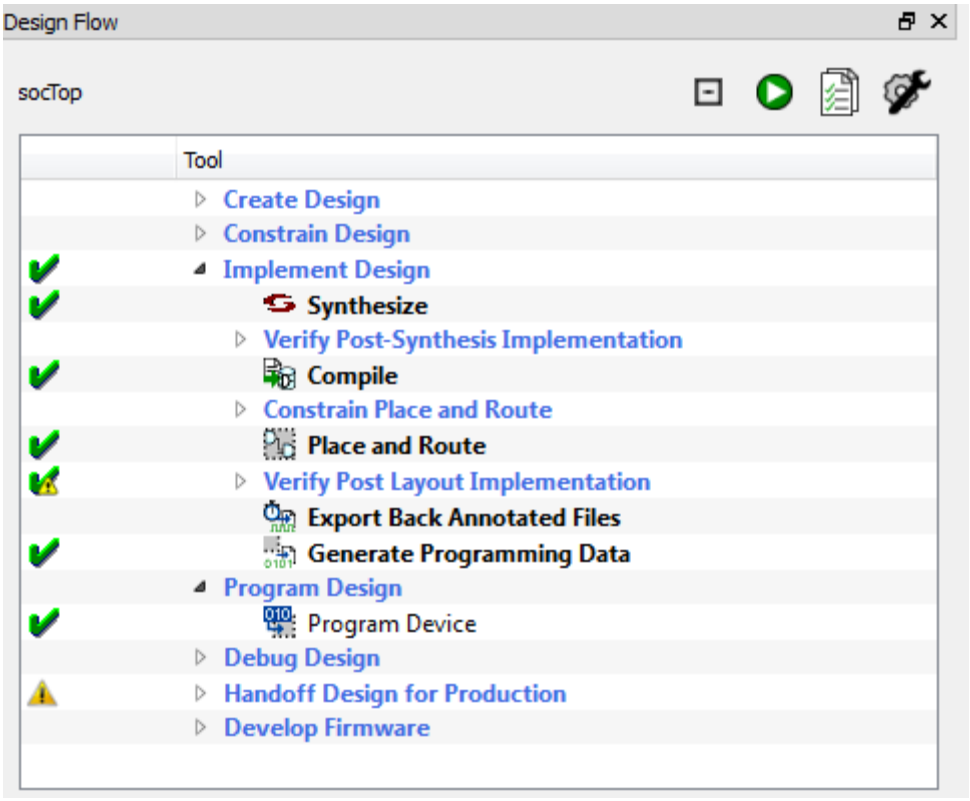


Figura26 Pasos de Implementación y programación

Los dos primeros (sintetizar y compilar) pasos son una compilación del SmartDesign dentro del proyecto global.

El siguiente paso sería hacer click en *Place &Route* tal y como muestra la Figura26. Sin embargo es imprescindible que despleguemos el menú de restricciones de este paso, ya que en él se encuentran las opciones para configurar las asignaciones físicas de las I/O y la configuración de tiempos. Si no configuramos estos aspectos es muy probable que aunque nuestro SmartDesign no contenga errores, el diseño no funcione correctamente. Esto es debido a que la asignación de muchos puertos I/O de propósito general se asignan automáticamente, de manera que pueden no ser los deseados o directamente no saber que puertos son.

Cuando desplegamos el menú, observamos tres opciones tal como se ve en la Figura27

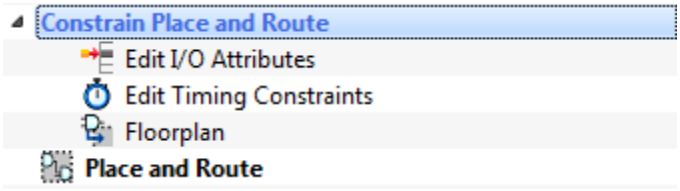
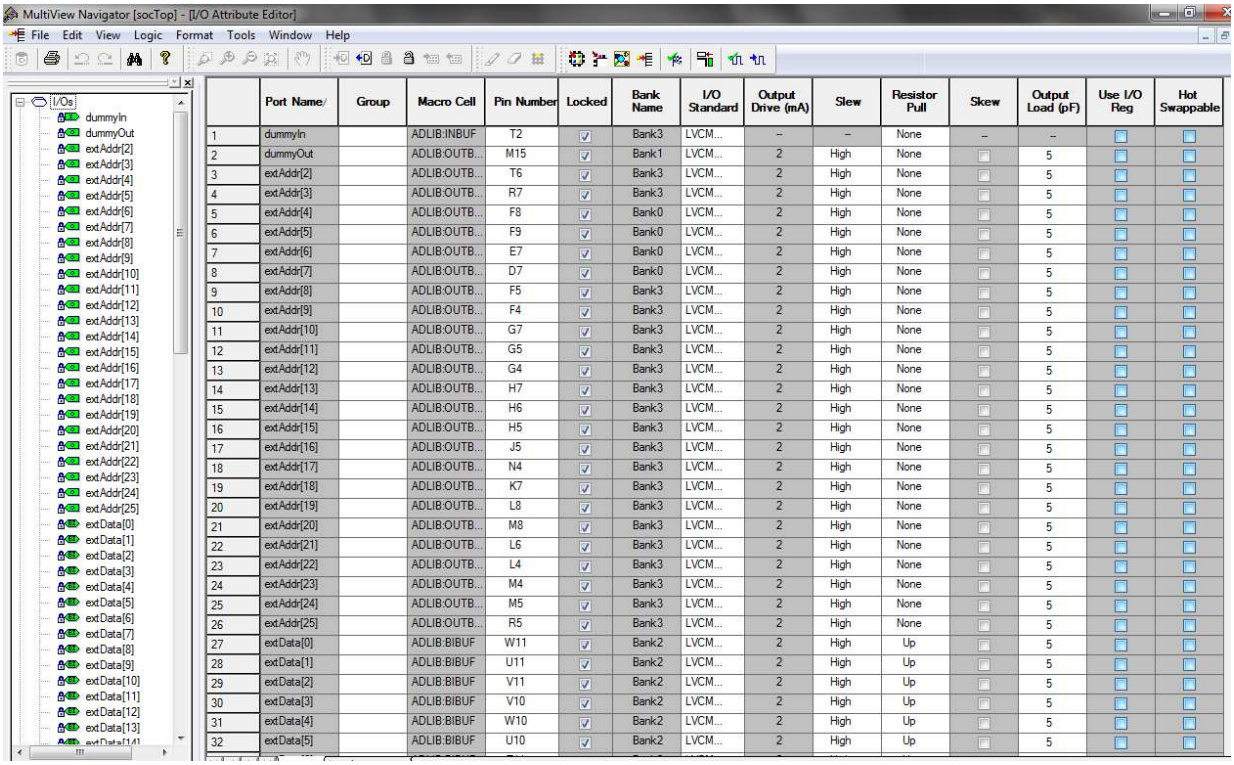


Figura27 Edición de I/O y tiempos

Comenzaremos editando la configuración de I/O. al hacer doble click en la herramienta de edición arranca un programa nuevo dentro del Libero SoC, llamado Designer, en el cual realizaremos los cambios. La interfaz a la que nos envía directamente al hacer click en esta opción es la que se muestra en la Figura28.



	Port Name/	Group	Macro Cell	Pin Number	Locked	Bank Name	I/O Standard	Output Drive (mA)	Slew	Resistor Pull	Skew	Output Load (pF)	Use I/O Reg	Hot Swappable
1	dummyIn		ADLIB:INBUF	T2	☑	Bank3	LVCM...	--	--	None	--	--	☐	☐
2	dummyOut		ADLIB:OUTB...	M15	☑	Bank1	LVCM...	2	High	None	☐	5	☐	☐
3	extAddr[2]		ADLIB:OUTB...	T6	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
4	extAddr[3]		ADLIB:OUTB...	R7	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
5	extAddr[4]		ADLIB:OUTB...	F8	☑	Bank0	LVCM...	2	High	None	☐	5	☐	☐
6	extAddr[5]		ADLIB:OUTB...	F9	☑	Bank0	LVCM...	2	High	None	☐	5	☐	☐
7	extAddr[6]		ADLIB:OUTB...	E7	☑	Bank0	LVCM...	2	High	None	☐	5	☐	☐
8	extAddr[7]		ADLIB:OUTB...	D7	☑	Bank0	LVCM...	2	High	None	☐	5	☐	☐
9	extAddr[8]		ADLIB:OUTB...	F5	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
10	extAddr[9]		ADLIB:OUTB...	F4	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
11	extAddr[10]		ADLIB:OUTB...	G7	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
12	extAddr[11]		ADLIB:OUTB...	G5	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
13	extAddr[12]		ADLIB:OUTB...	G4	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
14	extAddr[13]		ADLIB:OUTB...	H7	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
15	extAddr[14]		ADLIB:OUTB...	H6	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
16	extAddr[15]		ADLIB:OUTB...	H5	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
17	extAddr[16]		ADLIB:OUTB...	J5	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
18	extAddr[17]		ADLIB:OUTB...	N4	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
19	extAddr[18]		ADLIB:OUTB...	K7	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
20	extAddr[19]		ADLIB:OUTB...	L8	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
21	extAddr[20]		ADLIB:OUTB...	M8	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
22	extAddr[21]		ADLIB:OUTB...	L6	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
23	extAddr[22]		ADLIB:OUTB...	L4	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
24	extAddr[23]		ADLIB:OUTB...	M4	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
25	extAddr[24]		ADLIB:OUTB...	M5	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
26	extAddr[25]		ADLIB:OUTB...	R5	☑	Bank3	LVCM...	2	High	None	☐	5	☐	☐
27	extData[0]		ADLIB:BIBUF	W11	☑	Bank2	LVCM...	2	High	Up	☐	5	☐	☐
28	extData[1]		ADLIB:BIBUF	U11	☑	Bank2	LVCM...	2	High	Up	☐	5	☐	☐
29	extData[2]		ADLIB:BIBUF	V11	☑	Bank2	LVCM...	2	High	Up	☐	5	☐	☐
30	extData[3]		ADLIB:BIBUF	V10	☑	Bank2	LVCM...	2	High	Up	☐	5	☐	☐
31	extData[4]		ADLIB:BIBUF	W10	☑	Bank2	LVCM...	2	High	Up	☐	5	☐	☐
32	extData[5]		ADLIB:BIBUF	U10	☑	Bank2	LVCM...	2	High	Up	☐	5	☐	☐

Figura28 Edición de I/O

En esta tabla disponemos de toda la información de la asignación de pines de la FPGA con inputs y outputs del Smart Design.

Para poder configurar algunas conexiones es necesario mirar la asignación en la documentación de la placa en su datasheet, como por ejemplo saber qué salidas del microprocesador están unidas a los LEDs.

En este documento podemos encontrar varias tablas con asignaciones como las siguientes (Figura29) para el modelo con encapsulado GF484 que es el que nos interesa:

FG484 Ball	Preloaded Design Signal	Schematic Signal	I/O	Development Kit Function
T9	pbRstN	BUF2_PBRST_N	I	Push Button Reset
V7	poRstN	PORESET_N	I	Power on Reset
T7	flashRstN	FLASH_RST_N	O	Reset to Flash Chips
E4	extClk	OSC_CLK	I	System Clock
D9	rs232Arx	RS232_RX	I	UART Receive
G9	rs232Atx	RS232_TX	O	UART Transmit
T6	memAddr[2]	MEM_ADDR2	O	SRAM / flash address
R7	memAddr[3]	MEM_ADDR3	O	SRAM / flash address
F8	memAddr[4]	MEM_ADDR4	O	SRAM / flash address
F9	memAddr[5]	MEM_ADDR5	O	SRAM / flash address
E7	memAddr[6]	MEM_ADDR6	O	SRAM / flash address
D7	memAddr[7]	MEM_ADDR7	O	SRAM / flash address
F5	memAddr[8]	MEM_ADDR8	O	SRAM / flash address
F4	memAddr[9]	MEM_ADDR9	O	SRAM / flash address
G7	memAddr[10]	MEM_ADDR10	O	SRAM / flash address
G5	memAddr[11]	MEM_ADDR11	O	SRAM / flash address
G4	memAddr[12]	MEM_ADDR12	O	SRAM / flash address
H7	memAddr[13]	MEM_ADDR13	O	SRAM / flash address
H6	memAddr[14]	MEM_ADDR14	O	SRAM / flash address
H5	memAddr[15]	MEM_ADDR15	O	SRAM / flash address
J5	memAddr[16]	MEM_ADDR16	O	SRAM / flash address
N4	memAddr[17]	MEM_ADDR17	O	SRAM / flash address
K7	memAddr[18]	MEM_ADDR18	O	SRAM / flash address
L8	memAddr[19]	MEM_ADDR19	O	SRAM / flash address
M8	memAddr[20]	MEM_ADDR20	O	SRAM / flash address
L6	memAddr[21]	MEM_ADDR21	O	SRAM / flash address
L4	memAddr[22]	MEM_ADDR22	O	SRAM / flash address
M4	memAddr[23]	MEM_ADDR23	O	SRAM / flash address
M5	memAddr[24]	MEM_ADDR24	O	SRAM / flash address
R5	memAddr[25]	MEM_ADDR25	O	SRAM / flash address
W11	memData[0]	MEM_DATA0	I/O	SRAM / flash data
U11	memData[1]	MEM_DATA1	I/O	SRAM / flash data
V11	memData[2]	MEM_DATA2	I/O	SRAM / flash data
V10	memData[3]	MEM_DATA3	I/O	SRAM / flash data

Figura29 Asignación puertos I

De manera que estudiando la asignación de los pines que nos interesan, simplemente tenemos que cambiarlos en la herramienta de la Figura28.

Una vez hemos configurado todas las entradas y salidas podemos acceder al siguiente menú, edición de restricciones de tiempos (Figura30).

En esta herramienta vamos a controlar la frecuencia del reloj, así como la de los dispositivos de comunicación. Tal como se muestra en la Figura30, el cambio de los valores es muy sencillo, basta con hacer click en el número y escribir el nuevo valor.

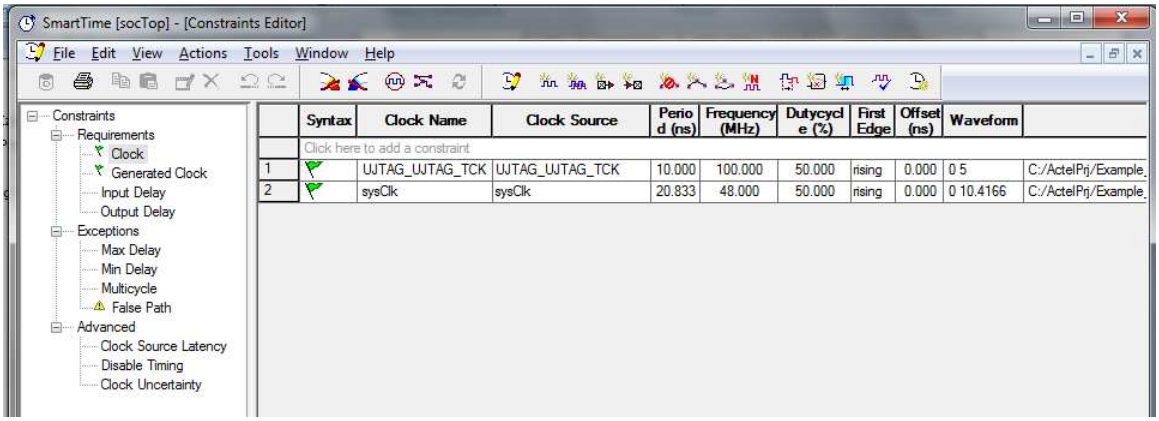


Figura30 Edición de tiempos

La última tarea que hay en este apartado es Floorplant, en esta herramienta no vamos a configurar nada pero nos permite ver el espacio ocupado por el diseño así como las conexiones con los pines (Figura31).

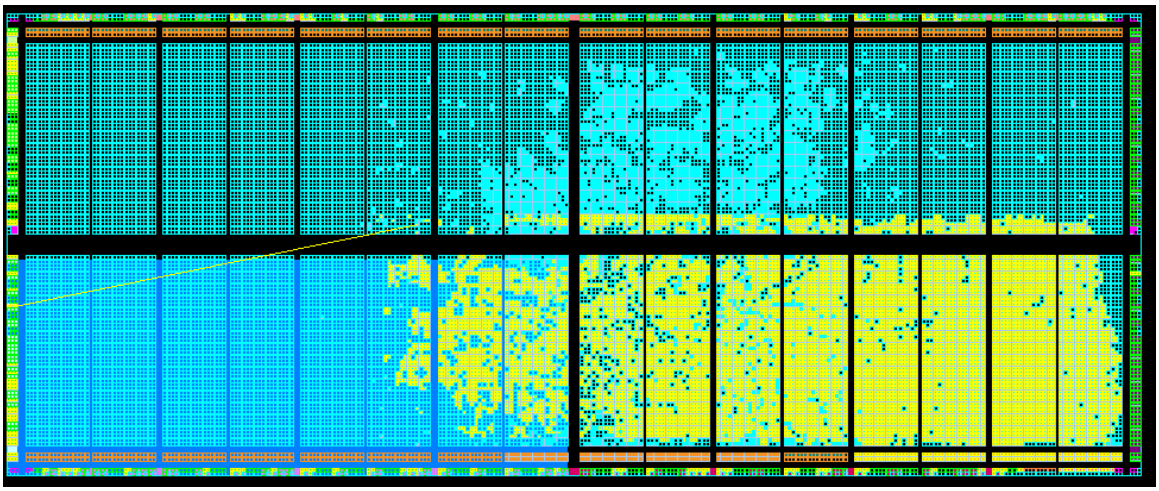


Figura31 Mapa de la FPGA

El siguiente punto los pasos de ejecución del “DesignFlow” es el “Place & Route”, esta configuración tarda un tiempo significativamente más elevado que el resto de pasos, se recomienda tener todos los pasos previos revisados y bien configurados con el fin de evitar esperas innecesarias en la ejecución de esta tarea.

Después de este punto podemos programar directamente la placa.

Una vez hemos hecho click en programar, esperaremos a que el LED amarillo de la placa deje de parpadear y el programa avisará de que la placa se ha configurado correctamente. Hasta este paso no se ha necesitado de la comunicación entre placa y PC. Si al programarla sucede algún error de conexión se recomienda repasar las conexiones descritas al principio de este capítulo.

El diseño de hardware se ha finalizado. Lo último que nos queda hacer en el Libero SoC es llamar a la última herramienta del "DesignFlow" como muestra la Figura32.

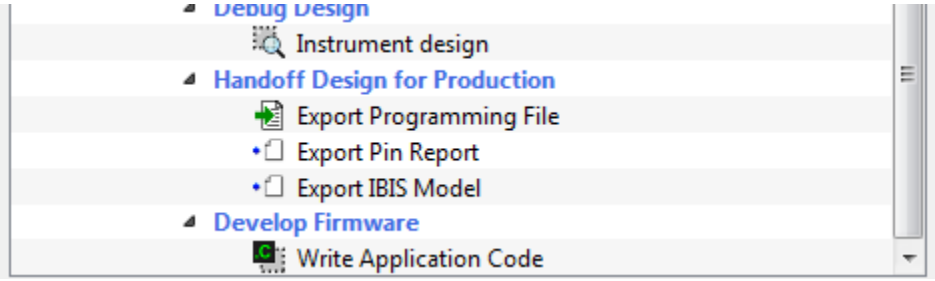


Figura32 Enlace al SoftConsole

Al hacer doble click en "WriteApplicationCode" se nos abre la herramienta de Actel encargada del diseño Software del sistema.

4.3 Tutorial para la herramienta SoftConsole

Esta herramienta nos permite manejar toda la edición relacionada con la parte software del diseño. Podemos elegir distintos tipos de lenguaje para el proyecto como, C o C++.

A la hora de realizar esta parte del proyecto, podemos elegir dos opciones:

- La primera es realizar el proyecto desde cero. Creando nuestro main.c, agregando las bibliotecas necesarias, enlazando las direcciones, etc.
- La segunda opción es importar un proyecto existente que tenga una arquitectura de diseño parecida a la que buscamos y en el cual las bibliotecas necesarias estén todas incluidas.

Ambas opciones son perfectamente validas y cada una tiene sus pros y sus contras.

El diseño propio desde cero nos permite tener un conjunto de archivos específicamente seleccionados para nuestra función, en el cual no habrá ningún archivo que no se vaya a utilizar, además podemos encontrar más rápido errores al estar más familiarizados con el conjunto de bibliotecas. Sin embargo esta opción tiene un problema, el linker. Estos problemas se comentaran más adelante en la opción del diseño desde cero.

La opción de importar un proyecto existente nos elimina o facilita considerablemente una parte del diseño, la arquitectura de archivos. La colocación de las librerías puede ocasionar problemas de direccionamiento si no se tiene extremado cuidado. Este problema no existe si utilizamos este método. El inconveniente es la poca flexibilidad a la hora de realizar una aplicación distinta a la que tenemos por defecto.

A continuación se exponen unos breves tutoriales, en los que se facilitará información sobre los pasos a seguir para realizar un proyecto.c, añadir un proyecto.c y realizar una depuración tanto con la herramienta del SoftConsole como en GDB.

4.3.1 Realización del proyecto desde cero

Lo primero que solicita el SoftConsole nada mas ejecutarlo es el “workspace” donde vamos a guardar nuestro proyecto, tal y como muestra laFigura33.

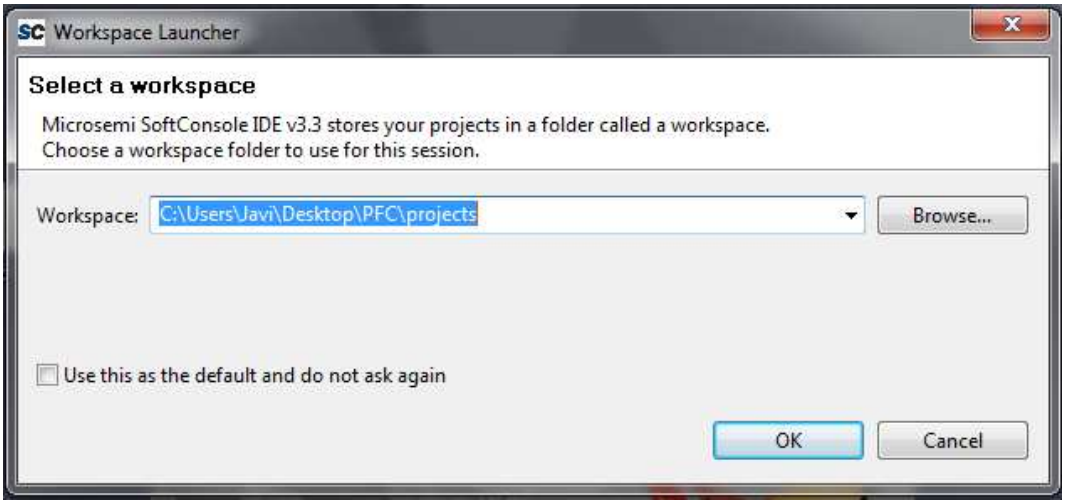


Figura33 Selección del Workspace

Como en todos los casos de selección de una dirección para guardar archivos, es recomendable elegir una dirección que no disponga de espacios, una válida seria la que se muestra en la Figura33.

Una vez abierta la herramienta, nos dirigimos a la esquina superior izquierda y desplegamos la pestaña File>New>C Project. Se nos desplegará un cuadro como el que se muestra en la Figura34. En el cual, introduciremos un nombre para nuestro proyecto, tendremos el "workspace" por defecto asignado, elegiremos el tipo de proyecto (ejecutable) y el tipo de microprocesador sobre el que se va a ejecutar. Tras seleccionar estas opciones se crea una nueva carpeta en la parte izquierda de la interfaz con el nombre de nuestro proyecto. Esta carpeta está vacía y en ella tendremos que incluir todas las librerías y archivos necesarios.

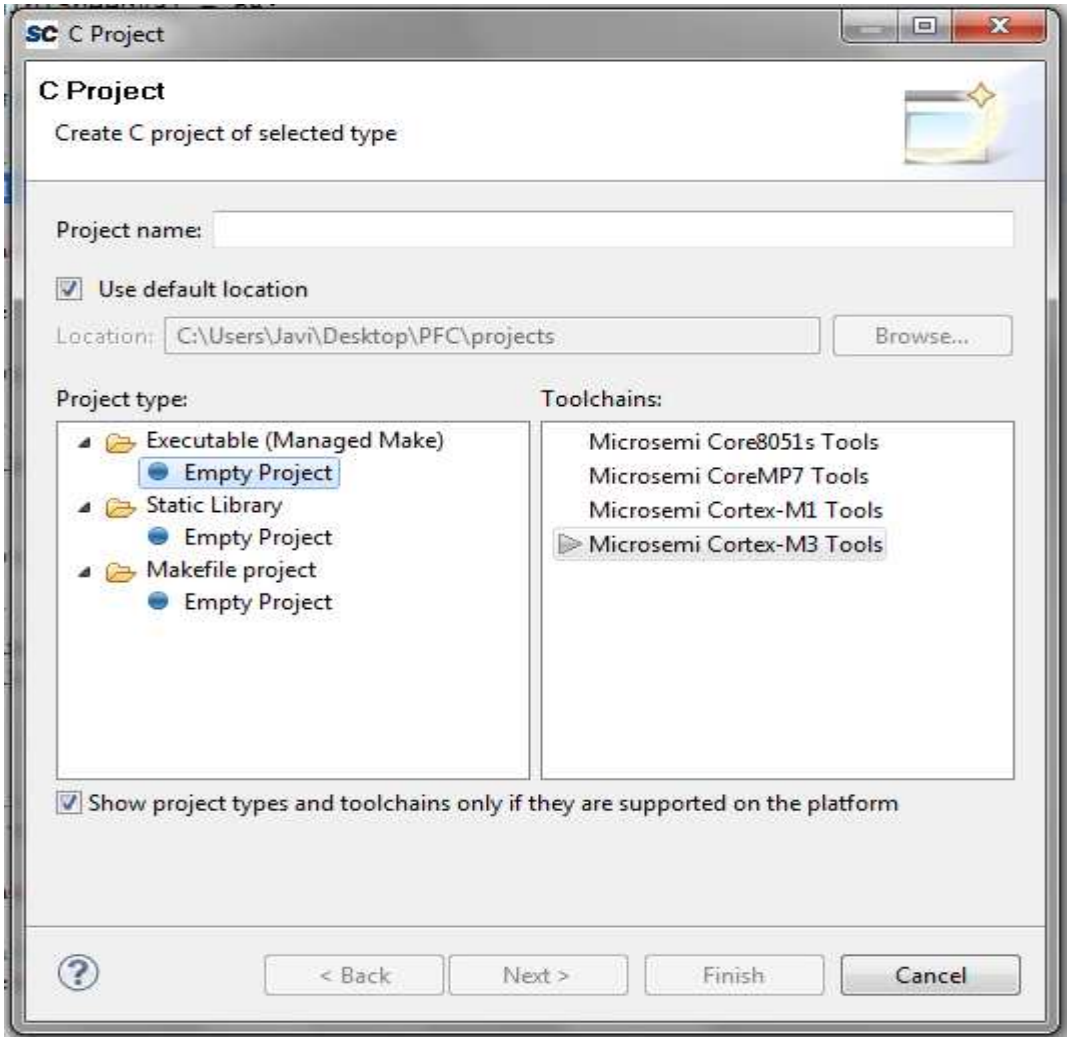


Figura34 Nuevo proyecto en C

El primer archivo a crear será nuestro archivo main, en el cual tendremos la aplicación a realizar por el microprocesador. Para crear este archivo basta con encontrar en la barra de herramientas superior un icono blanco con la letra C en azul. Tal como se ve en la

Figura35, hay distintas herramientas con la letra C, pero solo una con las características mencionadas anteriormente. Una vez hemos hecho click en la herramienta se despliega una ventana en la que tendremos que escribir el nombre del archivo y añadir su terminación, es decir, el formato ".c".

Figura35 Herramientas del SoftConsole

Ya podemos empezar a programar en nuestro archivo ".c". Una prueba de ello es la Figura36, en la que se ve un ejemplo del resultado de los pasos seguidos hasta este momento, con un ejemplo de texto en el archivo main.

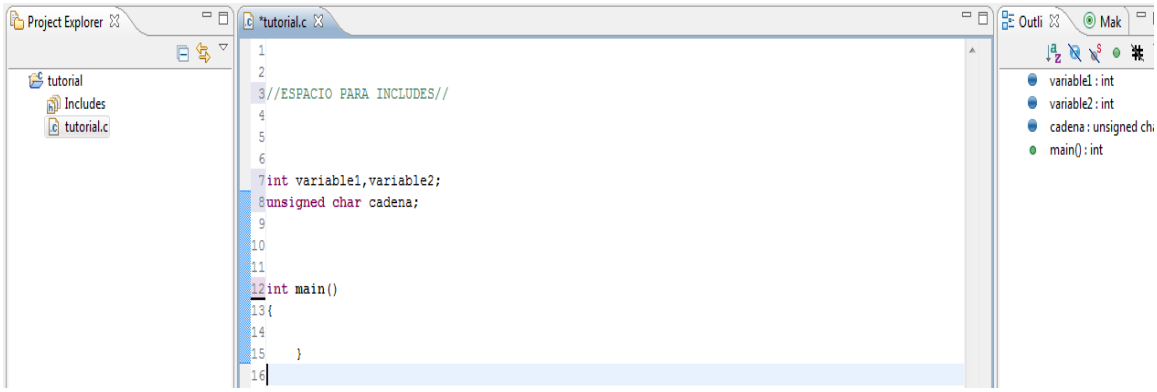


Figura36 Ejemplo de programa con main creado

Es obvio que para realizar un proyecto relacionado con el microprocesador necesitaremos muchas librerías que manejan los periféricos de este, las cuales, no están incluidas en el proyecto en un inicio. SoftConsole tiene una utilidad para importar bibliotecas de un servidor.

Esta utilidad se ejecuta seleccionando el icono situado en quinto lugar (comenzando desde la izquierda) en la barra de herramientas mostrada en la Figura35.

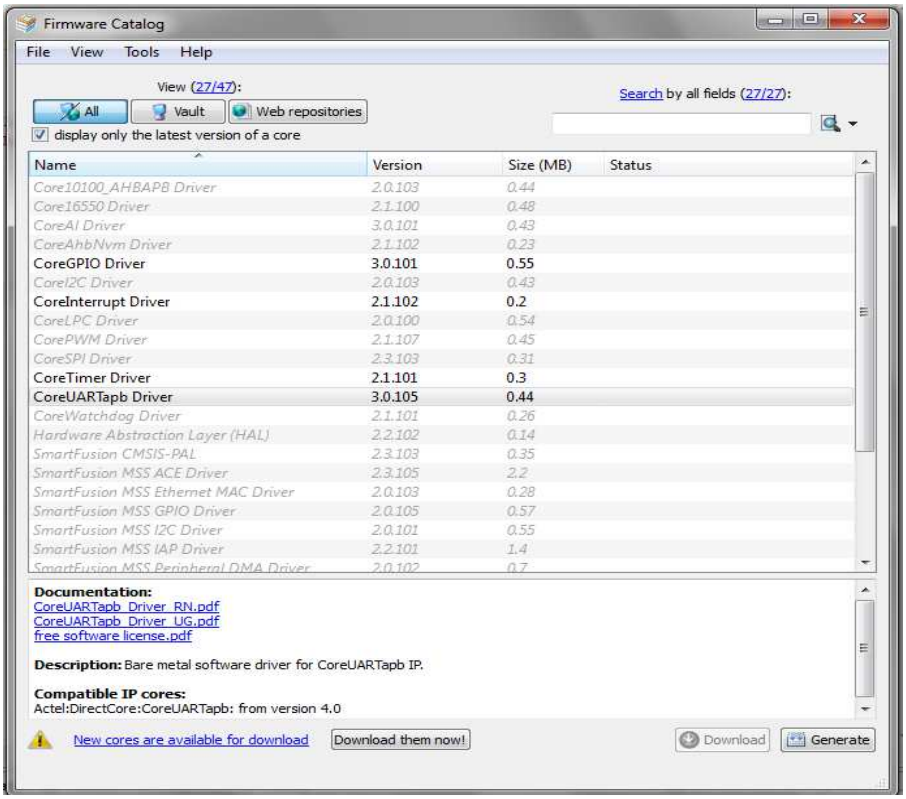


Figura37 Lista de bibliotecas

Tal como muestra la Figura37, disponemos de un amplio catalogo de bibliotecas para los distintos periféricos que pueda tener nuestro diseño hardware. Los que están en negrita son los que hemos descargado para nuestro ejemplo, para esto basta con hacer doble click sobre ellos, además si nos faltara información sobre cómo funcionan las funciones (las cuales vienen explicadas en los .h) abajo disponemos de links a documentos .pdf con información de cada una.

Una vez hemos descargado las bibliotecas necesarias, podemos observar cómo se han agregado a las lista de carpetas de la parte izquierda de la interfaz, dentro de “includes”, en “drivers” (Figura38).

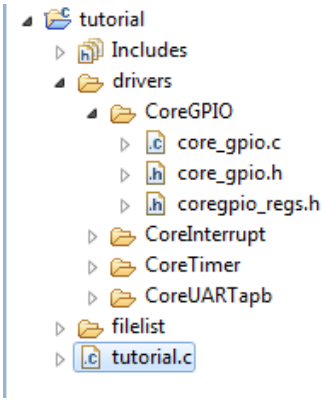


Figura38 Bibliotecas agregadas al proyecto

En estos packs de bibliotecas disponemos de archivos ".c" y ".h". En los archivos .c tenemos todas las funciones que podemos realizar con cada periférico detalladas con todo su código, mientras que en los .h disponemos de unos breves tutoriales de cómo funciona cada función y de las entradas que son necesarias para cada una.

Hay que resaltar dentro de las ".h" la necesidad de incluir la información de las direcciones de memoria de nuestra arquitectura hardware. Se encuentran definidas por la herramienta Libero SoC en la carpeta *firmware* del proyecto, en un archivo con el nombre de este acabado en *hw_platform.h*

Con este material ya seriamos capaces de realizar un proyecto completo. Simplemente bastaría con definir la configuración de tiempos a través del "coreTimer", la comunicación de la placa a través de "coreGPIO" y "coreUARTapb" y la ejecución de acciones a través de "coreInterrupt" (el tutorial de cómo utilizar estas funciones no es objeto de trabajo de este proyecto).

Antes de compilar es necesario incluir unas bibliotecas que seguramente no tengamos en el proyecto, las bibliotecas HAL (Hardware Abstraction Layer). Esto es debido a que las librerías que hemos descargado de los periféricos del sistema hacen referencia a estas. Estas librerías habilitan las interrupciones que realizan los periféricos y realizan funciones de bajo nivel. A diferencia de las anteriores la carpeta que se le asigna a estas bibliotecas es distinta por lo que debemos cambiarla antes de guardarla e incluirla con el resto de drivers.

Aun es necesario realizar una configuración previa a la compilación para evitar errores. Por defecto, el programa asigna la búsqueda de bibliotecas dentro de la carpeta *Includes*. En teoría las subcarpetas dentro de esta dirección están incluidas en la búsqueda pero suelen ocurrir errores con librerías ubicadas en subcarpetas. Lo más rápido para ubicar las librerías que van a causar problemas es intentar compilar y visualizar en la interfaz que ".h" o ".c" no han sido encontrados y proceder de la siguiente manera:

- Buscamos en la interfaz izquierda de la aplicación la carpeta que contiene al archivo que da error, hacemos click con el botón derecho y elegimos propiedades.
- Se abre una ventana como la que tenemos en la Figura39.

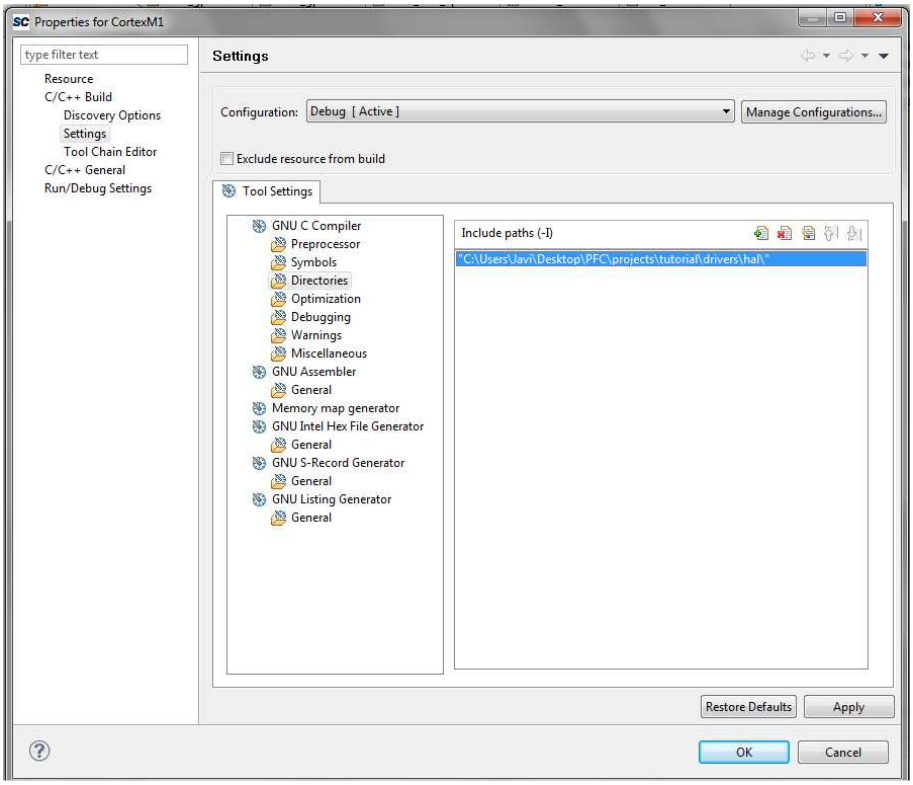


Figura39 Añadir dirección de bibliotecas

- Como se muestra en la Figura39 seleccionamos la opción de “Directories” y pulsamos el icono de agregar (cruz verde).
- Introducimos la dirección donde se encuentra el archivo que no reconoce (en este caso la carpeta en la que se contienen las HAL).
- Aplicamos cambios.

Una vez repitamos estos pasos para todos los archivos que den error podremos compilar el proyecto.

Una vez finalicemos nuestro proyecto siguiendo las instrucciones anteriores podemos encontrarnos con algún error sintáctico que tendremos que eliminar para poder completar la compilación.

Es posible que aun siguiendo estos pasos la compilación tenga errores, esto es debido a que el linker asignado no es el correcto.

El linker es un programa que toma los objetos generados en el proceso de compilación, la información de todos los recursos necesarios almacenados en bibliotecas, quita los recursos que no necesita y enlaza el código objeto produciendo un fichero ejecutable.

Para realizar estas funciones el linker debe poseer información sobre la distribución del mapa de memoria donde se va a implementar el programa, ya que necesita conocer las direcciones reservadas para el código.

Lo más lógico sería que la herramienta Libero SoC produjera un linker al finalizar el diseño, pero no hemos sido capaces de verificar su existencia.

Si no se consigue esta información se puede utilizar un linker que este diseñado para el mismo microprocesador ya que las direcciones serán las mismas casi en su totalidad.

Para cambiar el linker hacemos clik con el botón derecho en la carpeta del proyecto que se encuentra en la interfaz izquierda de la aplicación y seleccionamos propiedades.

Como se ve en la Figura40 disponemos de distintos campos de configuración del linker. Para saber el linker apropiado recomendamos consultar a Actel en (5).

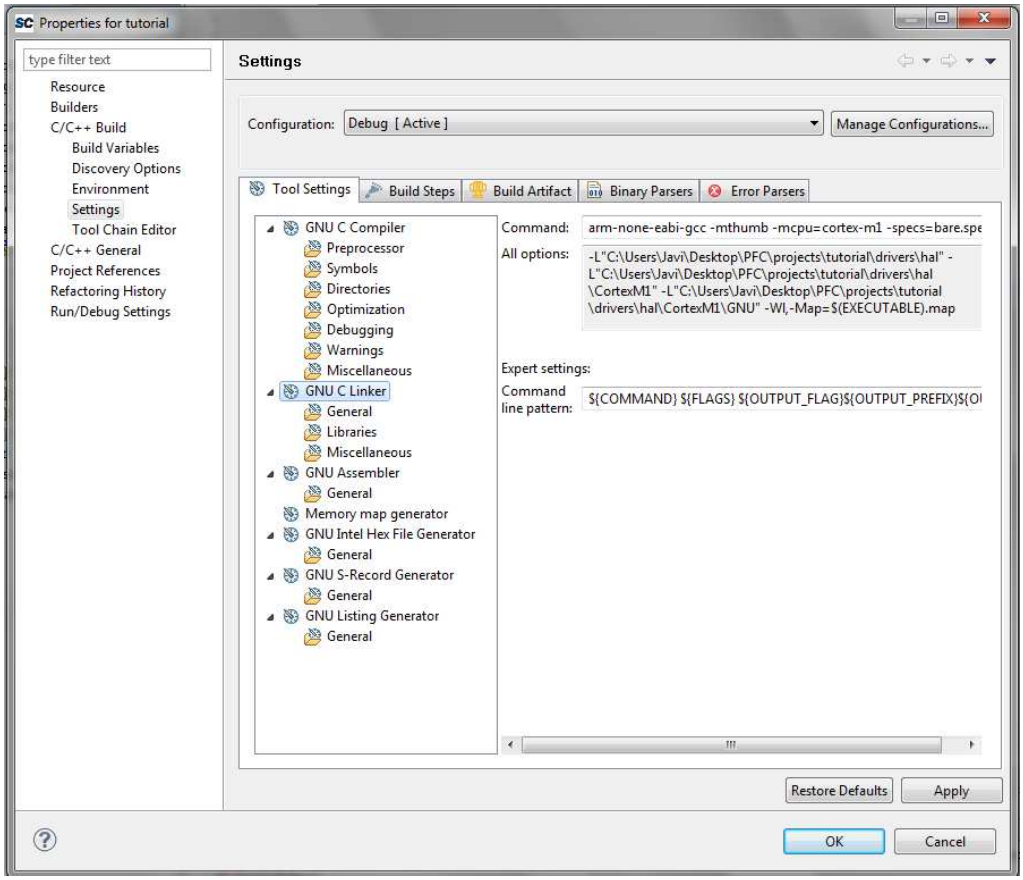


Figura40 Cambio del Linker

4.3.2 Realización de un proyecto desde otro ya existente

Para este caso dentro de la pestaña *file* seleccionamos la opción *import* y acedemos al menú que se muestra en la Figura41:

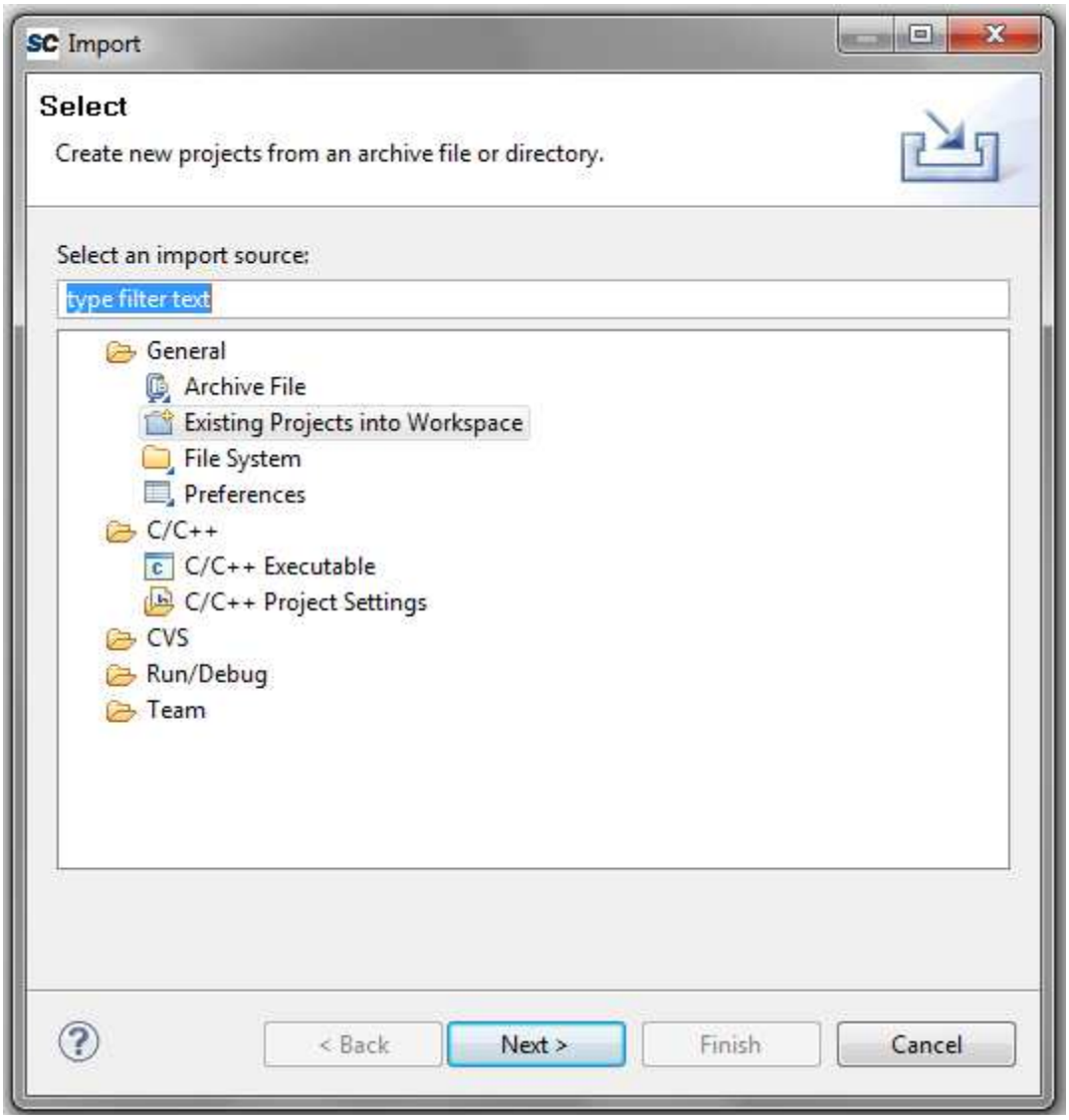


Figura41 Importar un proyecto en SoftConsole

Seleccionamos la opción marcada en la Figura41 y accedemos a una nueva ventana tal como vemos en la Figura42. En este menú buscamos la dirección donde tenemos ubicado el proyecto. Una vez hemos seleccionado la dirección tenemos que marcar solo el proyecto que vamos a querer importar y si nos interesa, marcar la opción de copiar al "workspace", y finalmente hacer click en "finish".

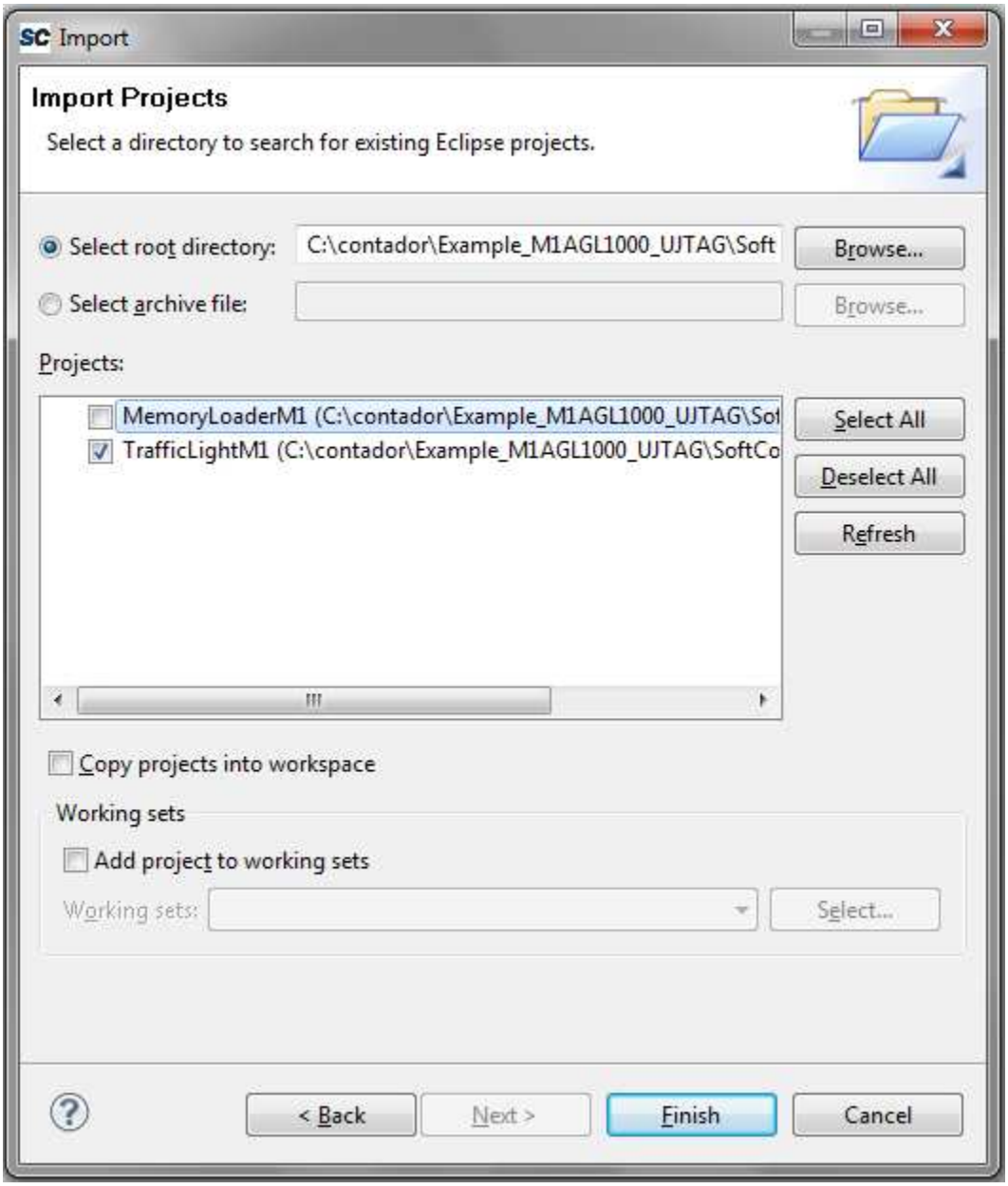


Figura42 Selección de proyecto a importar

Al finalizar este apartado, pasamos a tener un proyecto completo listo para compilar y ejecutar o depurar. En este caso el proyecto es la programación de un semáforo en el que van cambiando las luces (LEDs) en función del estado en el que se encuentra. Con el fin de que sea más fácil de detectar errores en nuestra aplicación hemos preferido cambiar este programa por uno que se encarga de realizar una cuenta en binario en los LEDs de manera que las luces permanecen el mismo tiempo encendidas en todos los estados y es mas intuitivo darse cuenta de que la cuenta no es la correcta a simple vista.

Cuando tengamos todo listo (como en la Figura43) estamos listos para compilar y ejecutar o depurar el proyecto.

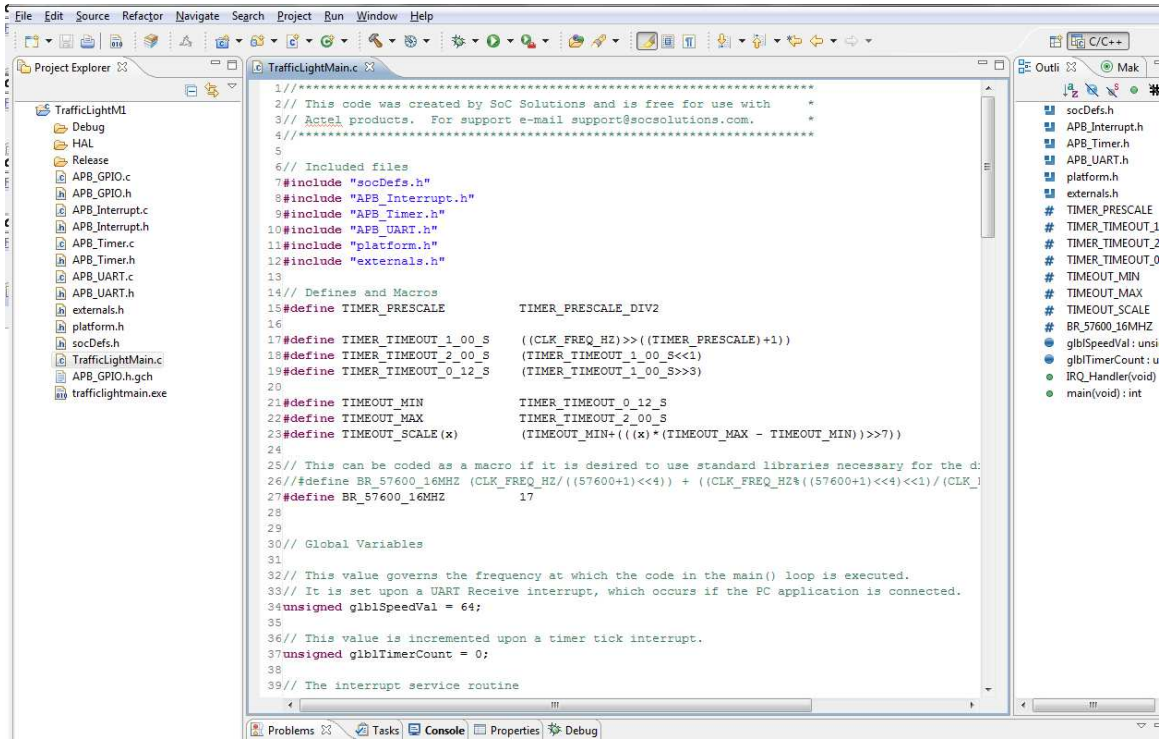


Figura43 Proyecto completo en SoftConsole

4.3.3 Depuración utilizando la herramienta SoftConsole

Una vez hemos compilado correctamente nuestro programa, pasaremos a explicar la herramienta de depuración. En este apartado explicaremos como realizar la depuración con la herramienta SoftConsole.

Partimos del punto en el que tenemos el programa recién compilado. Vamos a la carpeta del proyecto, hacemos click con el botón derecho y vamos a la opción “debug as” tal como muestra la Figura44.

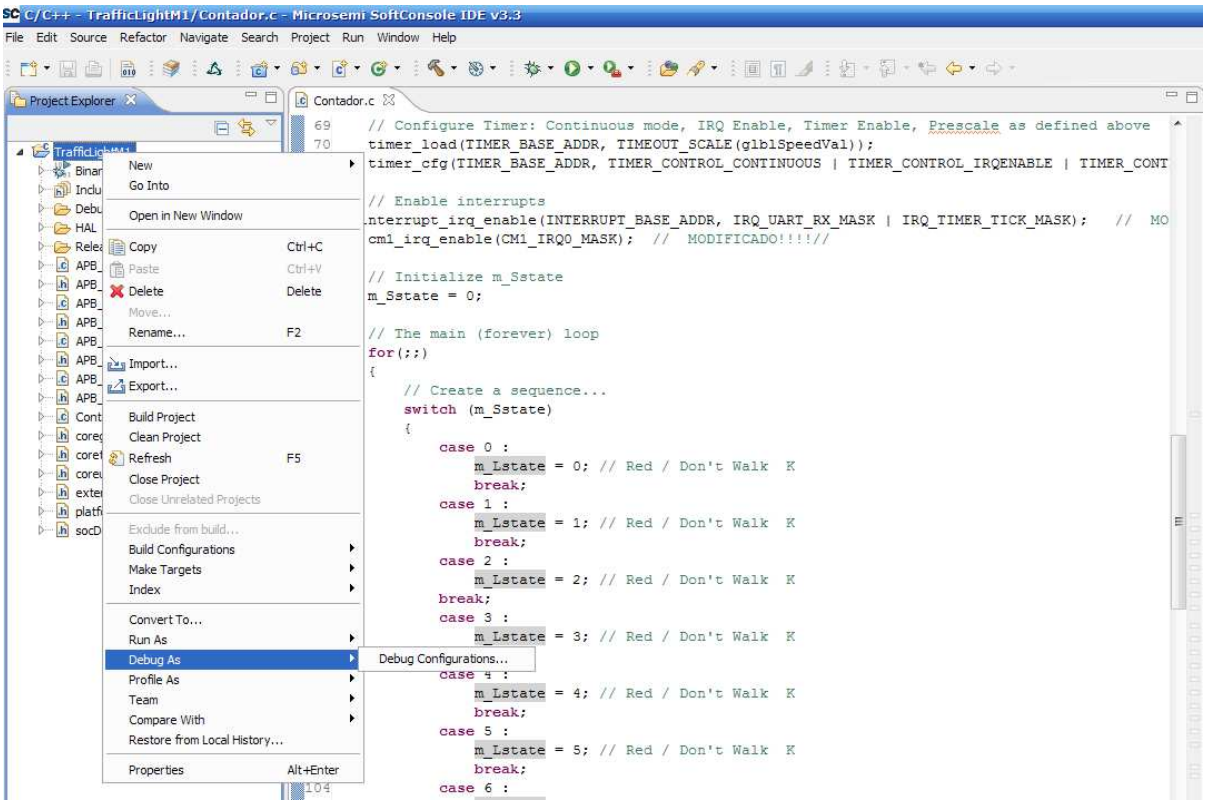


Figura44 Acceso a menú de depuración

Al hacer click en “DebugConfigurations” accederemos al menú de configuración previo a la depuración. En este menú, en primer lugar tenemos que seleccionar para qué tipo de microprocesador es la depuración, tal como muestra la Figura45.

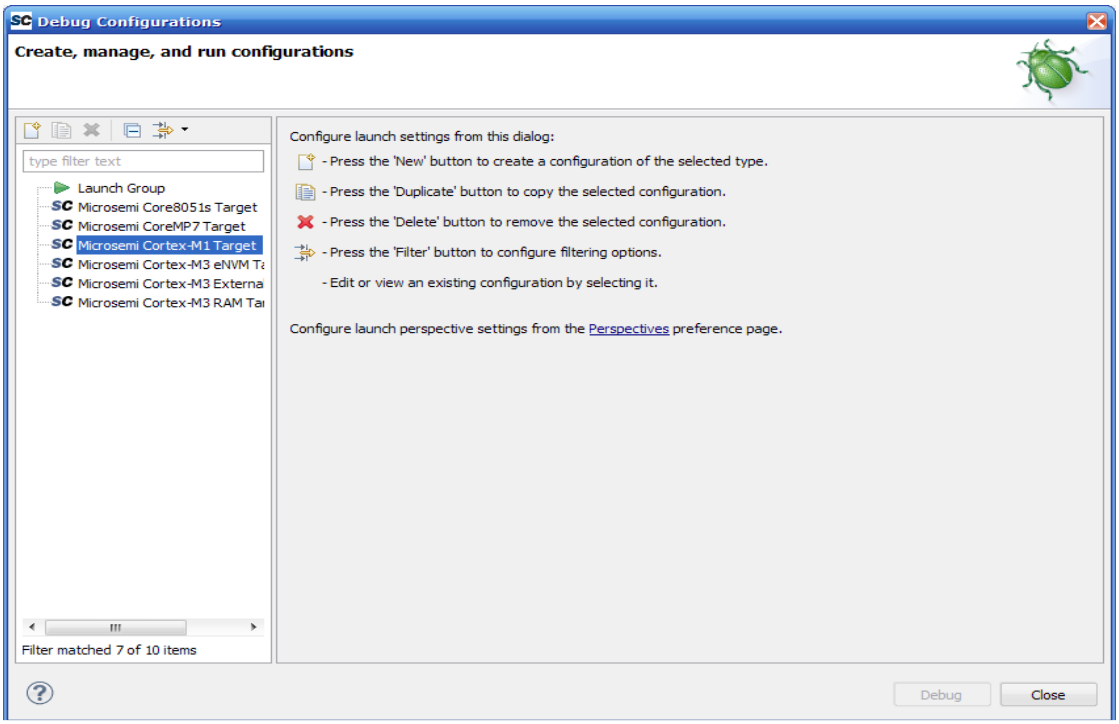


Figura45 Configuración de la depuración I.

Seleccionamos la opción Cortex M1, de manera que se nos genera una configuración para nuestro proyecto (si quisiéramos podríamos seleccionar otro). En esta ventana seleccionamos la opción “searchproject” y añadimos el que está en la carpeta “debug” tal y como muestra la Figura46.

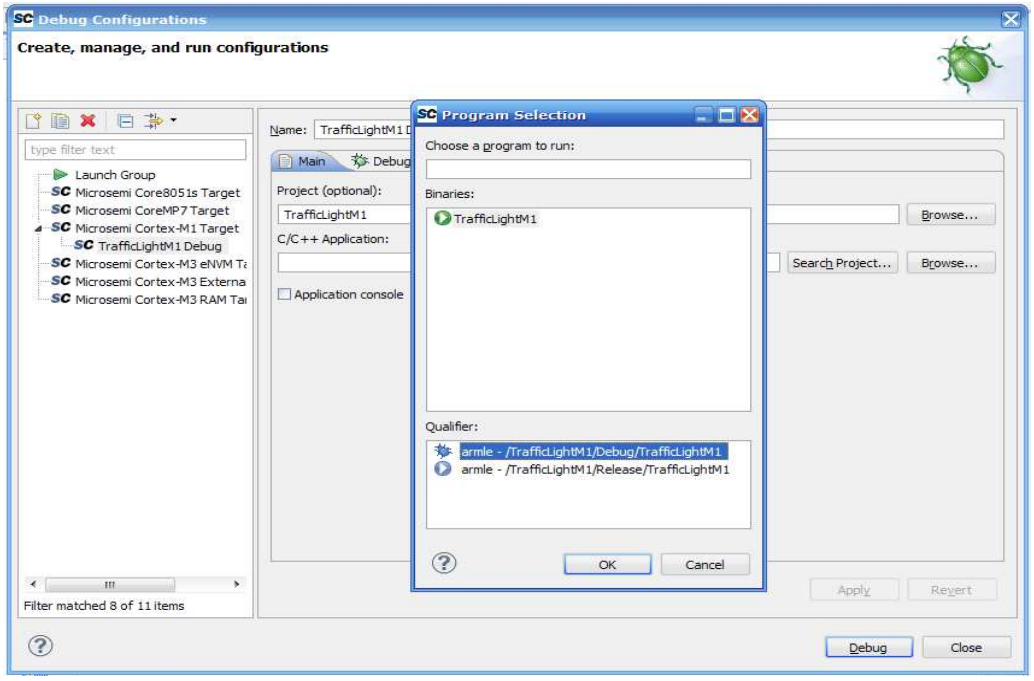


Figura46 Configuración de la depuración II.

Después de esta configuración previa ya estamos listos para lanzar la depuración seleccionando la pestaña de “Debug”.

Al lanzar la depuración la interfaz del SoftConsole cambia a la que se ve en la Figura47.

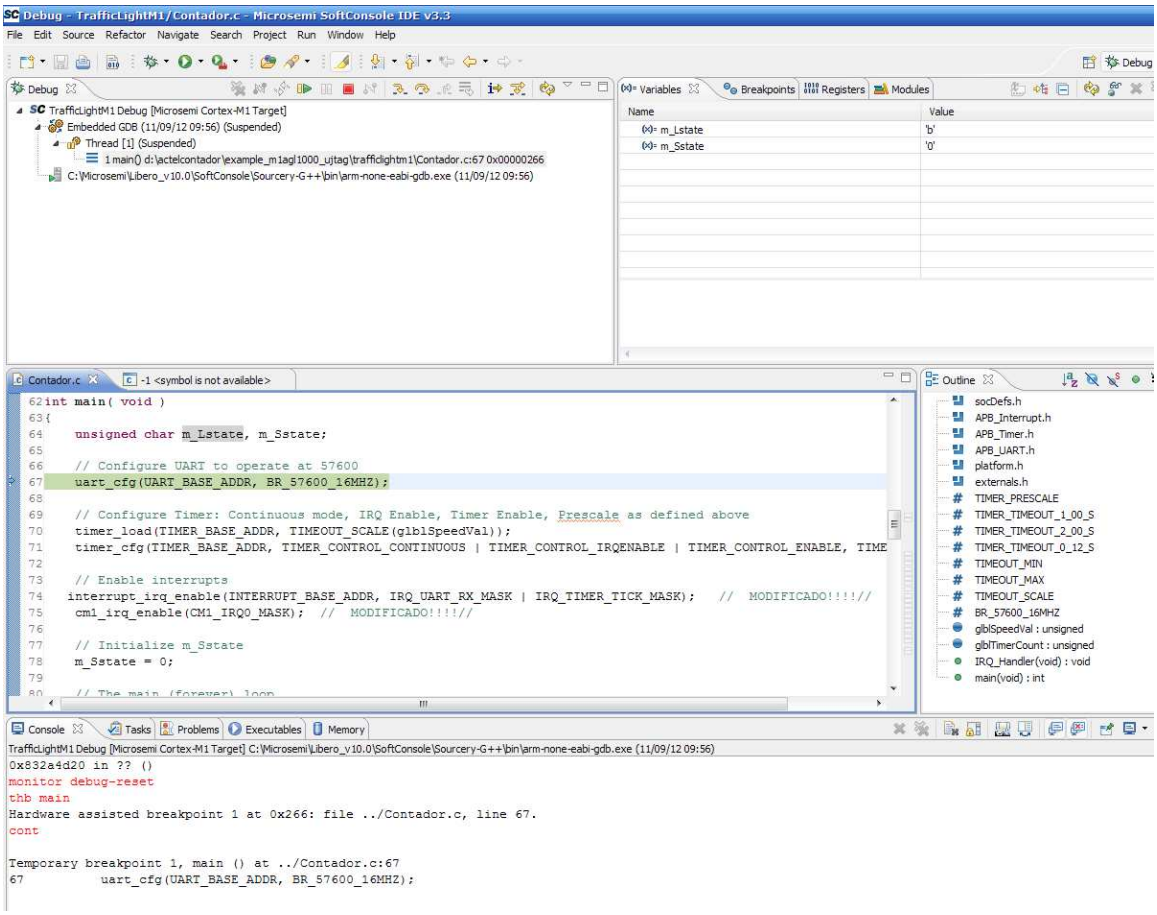


Figura47 Entorno de depuración

A continuación se describen las distintas partes de la herramienta:

- En la parte superior izquierda podemos apreciar que aplicación está utilizando el SoftConsole, en este caso vemos que es el GDB y en con que programa está siendo utilizado.
- En la parte central tenemos el código y un índice de variables y direcciones de posible interés.
- En la parte inferior tenemos un abanico de pestañas entre las que destacan:
    - la consola de ejecución. Encargada de comunicarse con el usuario a medida que se van ejecutando tareas.

- El reporte de errores. Donde podemos observar que ha fallado en un determinado paso.
- Buscador de valores de direcciones de memoria. En esta pestaña podemos visualizar zonas del mapa de memoria.

- Por último en la parte superior derecha tenemos las herramientas más útiles para la depuración:
  - Lista de variables del programa. En la que podemos alterar el valor de estas.
  - Lista de los registros de propósito general y configuración del sistema que posee el sistema. También modificables por el usuario.
  - Lista con los breakpoints activos y un índice con todos los archivos que forman el proyecto.

Al lanzarse la depuración, el propio programa inserta un breakpoint al inicio para detener el programa y permitirnos configurar breakpoints o alterar valores de inicio de registros, direcciones de memoria o variables.

Como podemos observar en la Figura48, esta herramienta es muy cómoda para alterar el valor de algún registro (igual para variables).

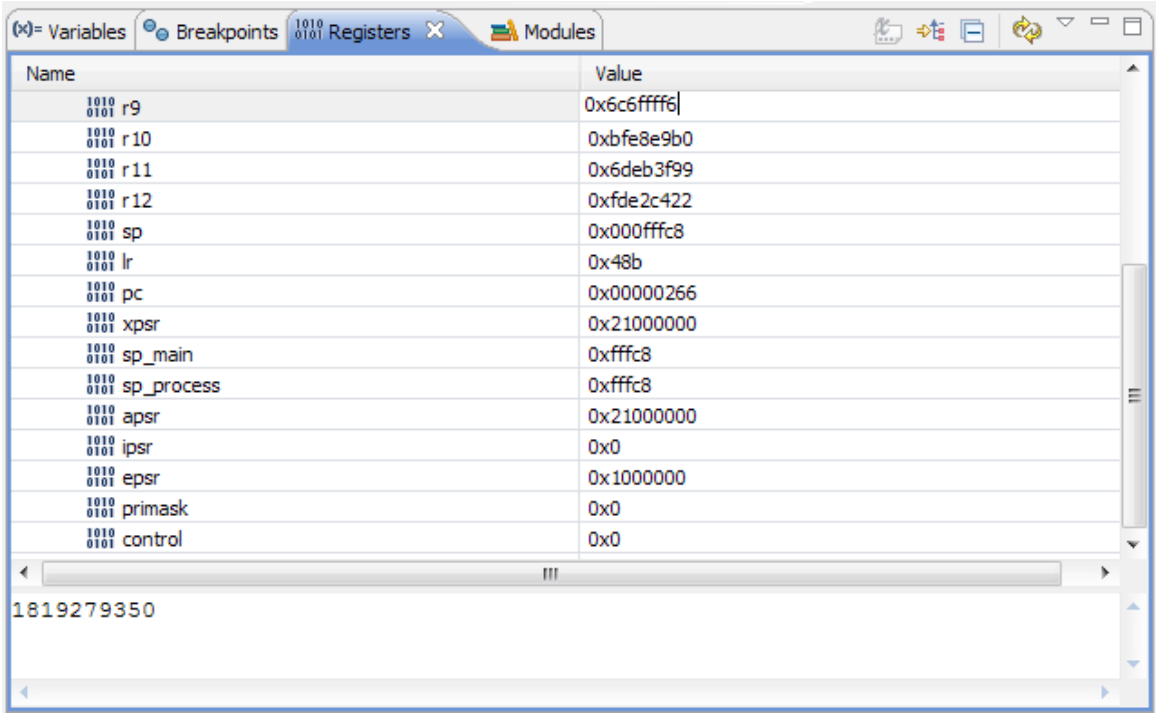


Figura48 Lista de registros modificables

Aunque esta aplicación resulta fácil e intuitiva para realizar cambios, no pose el aspecto fundamental de una inserción programa de fallos, es decir, la velocidad.

En una inserción programa de fallos se plantea inyectar una serie de fallos elevada.

De manera que si se implementase a mano con esta herramienta tardaríamos un tiempo muy elevado, por lo que este entorno de depuración no nos va a servir para nuestro fin.

Tendremos que buscar un entorno de comandos mas rápido (aunque sea menos intuitivo) que podamos implementar en un script, por ejemplo, de manera que ejecutando ese script comience toda la campaña de inserción de fallos programada.

4.4 Depuración utilizando la herramienta GDB

La herramienta GDB es una herramienta de depuración desarrollada como software libre. Esta herramienta de depuración es la que usa el SoftConsole a bajo nivel para realizar sus funciones de depuración.

Esto es debido a que el GDB está incluido en el programa que utiliza el Softconsole para todas las actividades de bajo nivel, como la compilación o la depuración del software, el *Sourcery G++ Lite*.

Esta herramienta software dispone de múltiples paquetes de compilación y depuración para distintas configuraciones de microprocesadores, entre ellos, ARM EABI (Embedded Application Binary Interface). La carpeta con todas las aplicaciones se encuentra dentro de la carpeta del SoftConsole creada en la instalación. Las aplicaciones se encuentran dentro de *sourcery-G++* (…*Sourcery-G++**bin*) como se muestra en Figura49. También disponemos de toda la información que necesitemos referente a el funcionamiento de cada uno de ellos (…*Sourcery-G++**share**doc**arm-none-eabi**pdf*). Esta aplicación es de software libre además la mayoría de las aplicaciones llevan incluido el prefijo *none,* que indica que funcionan con cualquier sistema operativo que tenga el microprocesador.

La finalidad de utilizar esta herramienta es, realizar las mismas funciones que realiza la herramienta SoftConsole en depuración, con la diferencia de realizarlas por comando para automatizar el proceso.

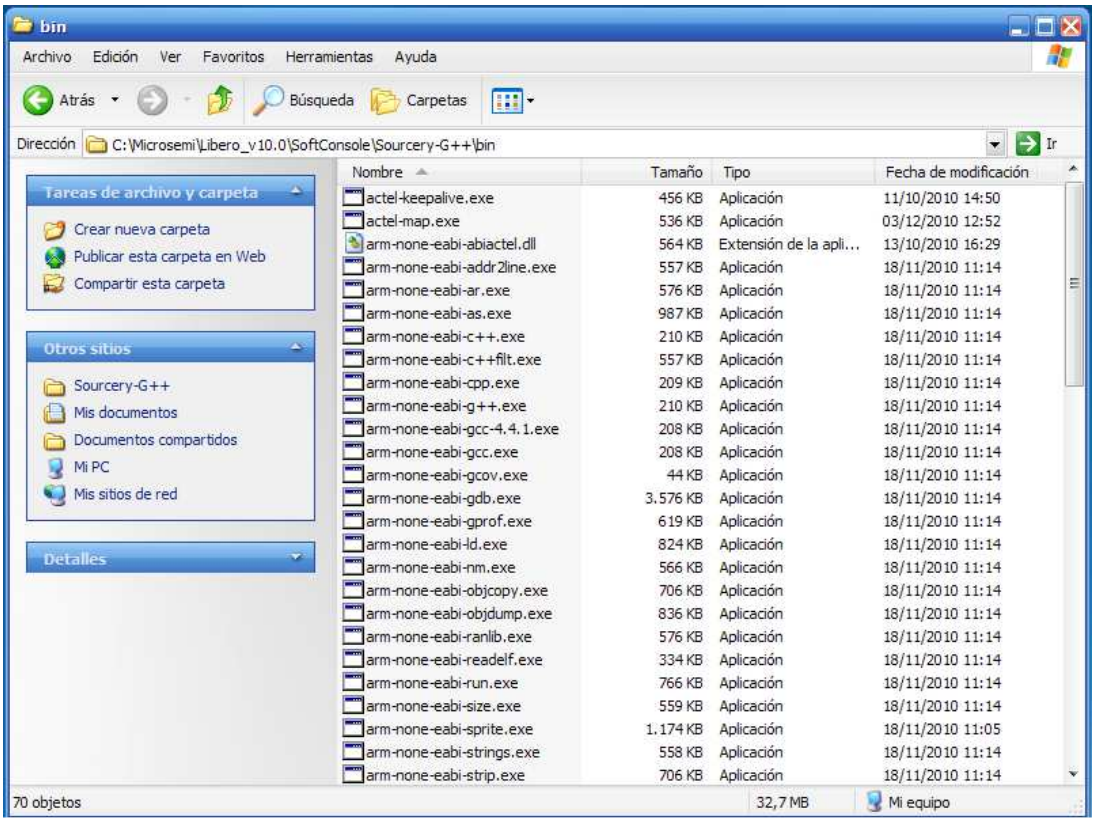


Figura49 Dirección de ejecutables del SoftConsole

Se explica a continuación como acceder y configurar la interfaz de depuración por comandos:

- Primero ejecutamos en la barra de inicio de Windows la consola CMD.
- El siguiente paso es dirigirnos a la carpeta *Sourcery-G++\bin*.
- Una vez en esta dirección, tal y como muestra la Figura50, introducimos el comando “startactel-keepaliveactel-keepalive”, el cual activa la comunicación con la FPGA. Para realizar este paso tenemos que tener la placa correctamente conectada al ordenador, se nos abrirá otra consola, en la cual no podemos escribir, es para recibir mensajes de la FPGA.

Figura50 Establecimiento de comunicación a través del CMD

Una vez creada la comunicación vamos a abrir el GDB, para ello tenemos que tener en cuenta que el programa ya este compilado de manera que tengamos un ejecutable al que acceder. Para ello, en la dirección mencionada anteriormente introducimos el comando que ejecuta la aplicación GDB dentro del Sourcery G++ y separado con un espacio, la dirección del programa que queremos depurar (Figura51):

Figura51 Lanzar el depurador

- Al realizar este paso ya hemos iniciado el depurador GDB. Tenemos que inicializar la configuración de la depuración a través de una serie de comandos:
    - *Set arm fallback-mode thumb*. Este comando controla el comportamiento del depurador. Al elegir el modo *thumb* le decimos al microprocesador que active el bit T del registro EPSR, estableciendo el tipo de instrucciones.
    - Una parte importante de la configuración es establecer que el programa que hemos seleccionado para depurar va a correr en la FPGA, para lo cual establecemos la comunicación con lo que es conocido como servidor externo de GDB. Esta comunicación se realiza a través de otro subprograma instalado dentro del SoftConsole, el *Eclipse*. De tal manera que ejecutamos el comando de GDB para dicho propósito, el cual

establecerá la configuración de comunicación con el periférico objetivo y le asignará el programa a depurar, como se expone a continuación:

-Target remote "F:/Microsemi/Libero_v10.0/SoftConsole/Eclipse//../Sourcery-G++/bin/arm-none-eabi-sprite"flashpro:?cpu=Cortex-M1 "Path_programa"

- Después de este paso el depurador nos devuelve unos comandos referentes a la configuración establecida, así como una dirección de memoria, la cual no posee información de interés. En este momento debemos introducir el siguiente comando para la configuración de la memoria, el cual hace que la memoria que no esté descrita explícitamente sea asignada como memoria RAM en vez de denegar el acceso a esa memoria:

  -set mem inaccessible-by-default off

- Estos pasos han configurado la comunicación y el entorno de la depuración pero aún no tenemos cargado el programa en la FPGA, para ello introducimos:

  -load

- Una vez hemos completado estos pasos estamos listos para realizar la depuración deseada. Para ello es conveniente conocer los comandos principales necesarios para depurar:
  - Run (r): ejecuta el programa.
  - Quit (q): Sale de la depuración.
  - Break + línea (b + línea): introduce un breakpoint en la línea deseada del programa.
  - Continue (c): continúa la ejecución después de un breakpoint.
  - Step (s): ejecuta paso a paso el código entrando en las funciones.
  - Next (n): ejecuta paso a paso el código sin entrar en funciones.

- Print /formato variable/dirección: lee la variable, registro o dirección de memoria en el formato indicado.
- Print variable/dirección= valor: asigna el valor deseado a una variable, registro o dirección de memoria.
- Print /formato $nombre: Lee un registro.
- Print /formato $nombre=valor: Escribe en un registro.
- X dirección: Accede a una dirección de memoria.
- Set {int} dirección: escribe en una dirección de memoria.

Como vemos, con esta manera de depurar, podemos realizar todas las opciones que nos permitía la aplicación del SoftConsole simplemente escribiendo comandos.

Esto nos permite que podamos crear un script en el que introduzcamos los comandos necesarios de depuración acoplados a un algoritmo, de manera que genere los errores deseados.

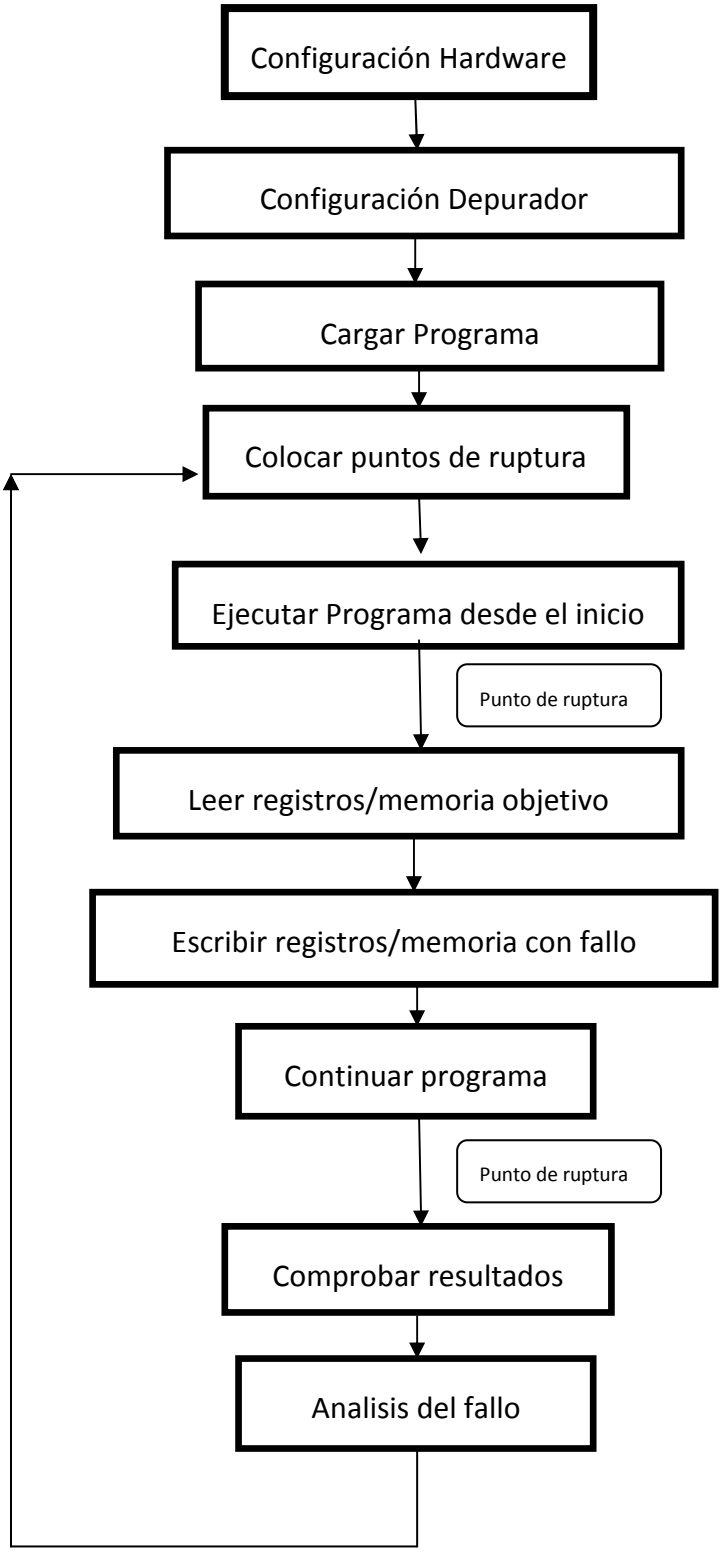
Capítulo 5

Pruebas Experimentales

En este capítulo vamos a hacer pruebas de depuración en el sistema integrado que hemos configurado a lo largo de todo el documento con el fin de poder responder a las preguntas que nos planteábamos al principio del proyecto, es decir, estudiar la viabilidad de un sistema software de inyección de fallos.

Las pruebas se han realizado en un programa que ha sido cargado en nuestro sistema embebido. Estas pruebas han servido para comprobar la viabilidad de inyectar fallos a través del depurador en modo comando. Para realizar las pruebas se han utilizado las herramientas de depuración que aporta la herramienta GDB.

El programa a ejecutar ha sido contador.c, el cual realiza una cuenta en binario a través de los LEDs que están en la placa. Las herramientas utilizadas principalmente del depurador GDB han sido los breakpoints y los comandos de lectura y escritura de registros y direcciones de memoria. A continuación se presenta un diagrama de flujo con los pasos realizados en las pruebas experimentales:


Configuración Hardware
Configuración Depurador
Cargar Programa
Colocar puntos de ruptura
Ejecutar Programa desde el inicio
Punto de ruptura
Leer registros/memoria objetivo
Escribir registros/memoria con fallo
Continuar programa
Punto de ruptura
Comprobar resultados
Analisis del fallo

Como se ha comentado anteriormente, el motivo de elección de este programa para el sistema integrado fue la facilidad de poder apreciar una alteración en el desarrollo del programa, ya que como se muestra en la Figura52 el seguimiento del programa puede hacerse observando los LEDs de la placa.

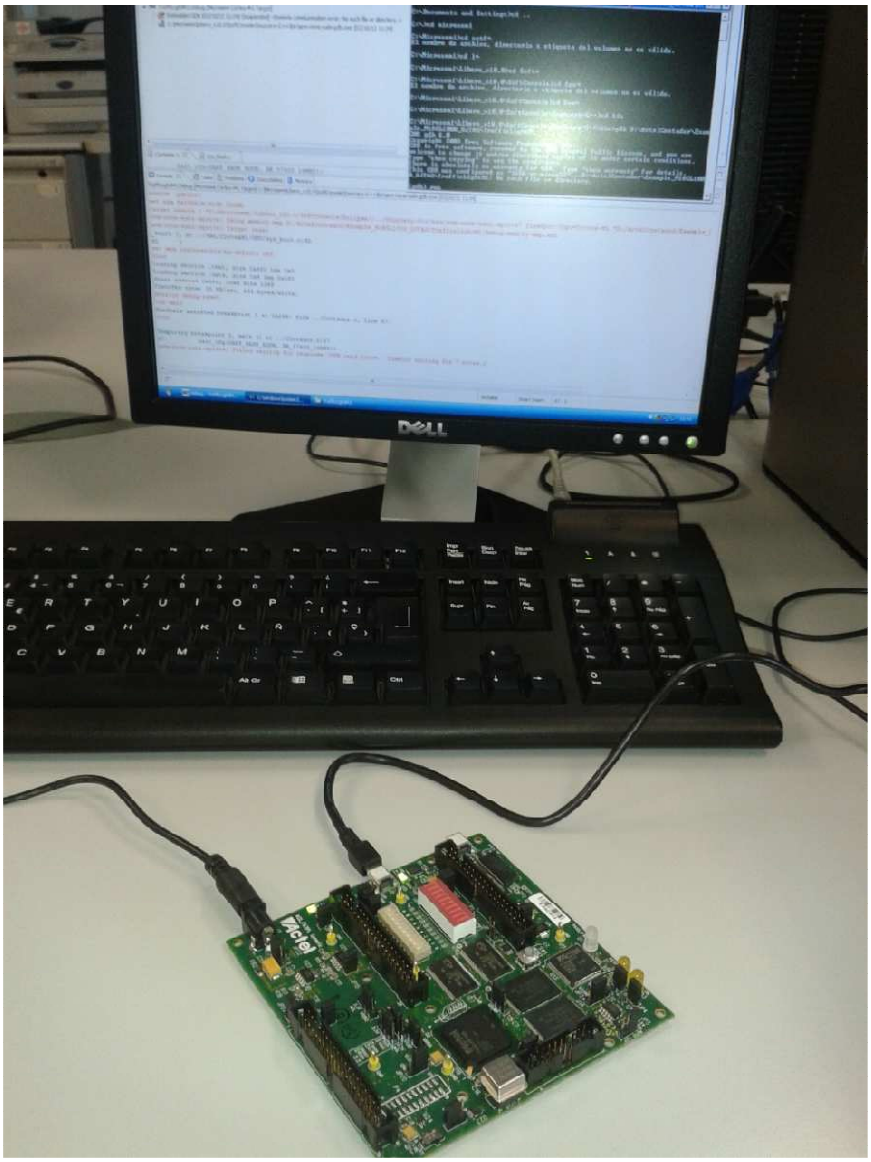


Figura52 Foto del sistema completo

Para comprobar que la depuración funciona, procederemos a alterar registros de propósito general, registros mapeados en memoria y memoria, observando las consecuencias, como mostramos a continuación:

Las primeras pruebas experimentales las realizamos en los registros de propósito general y de control del programa:

- El primer registro en el que hemos probado la inyección de fallos ha sido el PC (programcounter), también accesible con R15. Hemos podido leer un valor, modificarlo, comprobar que este era escrito y observar como el programa se veía afectado de distintas maneras dependiendo del valor inyectado y la situación de este.
- El siguiente registro ha sido el APSR, encargado del control de los flags del programa. Al escribir satisfactoriamente en este registro observamos cómo ocurrían errores sin más solución que cerrar la aplicación.
- Continuamos probando registros de control del programa. El IPSR, encargado de almacenar la excepción que está activa también ha sido accesible satisfactoriamente, recuperándose del error inyectado sin consecuencias en el programa.
- Para comprobar la posibilidad de inyección en registros que sean accesibles parcialmente incluimos el registro EPSR en las pruebas experimentales. Este registro dicta el modo de funcionamiento mediante el valor de un solo bit, siendo el resto del registro inaccesible. De manera que sabiendo la posición dentro del registro que ocupa el bit accesible, generamos un valor de registro que cambie solo ese bit. Al probarlo experimentalmente comprobamos que el valor del bit accesible se ha visto modificado.
- Después de probar con registros que controlan el sistema probamos en registros de propósito general. En todos conseguimos leer y escribir valores satisfactoriamente obteniendo distintas respuestas en función del estado de estos.

Tras haber realizado pruebas satisfactorias en la inyección de registros de propósito general y control del programa intentamos la inyección en registros direccionables por memoria.

- El primer registro probado es el encargado de escribir en los LEDs (registro asociado al GPIO). Inyectamos un valor distinto al almacenado y observamos como los LEDs cambian de estado acorde al valor introducido. La escritura es satisfactoria.

- Tras realizar esta escritura hemos procedido a leer el registro para comprobar que tenía el valor escrito (el cual se corresponde con el que muestran los LEDs). Sin embargo el valor leído no se corresponde con el escrito, por lo que nos encontramos con una incoherencia que nos indica que no podremos depurar en estos registros.

Para intentar solucionar este inconveniente procedemos a intentar leer el registro mediante software durante la ejecución en modo normal del programa. Sin embargo, tras varias pruebas el valor ha sido el mismo que el resultado obtenido mediante la lectura directa en GDB.

Por último (utilizando de nuevo GDB) probamos la lectura y escritura de una dirección de memoria accesible. El resultado tanto de lectura como de escritura es satisfactorio, ya que comprobamos con una lectura posterior que el valor había sido actualizado.

Los resultados obtenidos se resumen en la siguiente tabla:

Valores Modificados	Lectura	Escritura
Registros de propósito general	✓	✓
Registros de control	✓	✓
Registros mapeados en memoria	×	¿✓?
Direcciones de memoria	✓	✓

Tras estas pruebas experimentales realizadas estamos listos para poder sacar una conclusión sobre la viabilidad de la inyección programa de fallos.

Capítulo 6

Conclusiones y Trabajos futuros

6.1 Conclusiones

En este apartado procedemos a exponer las conclusiones y opiniones obtenidas a lo largo de todo el proyecto, tanto en la parte de creación del sistema como en las pruebas experimentales, observando los posibles fallos que puede tener el sistema, la capacidad de realizar una inyección de fallos en este tipo de dispositivos y la opinión que nos genera el modo de funcionamiento del sistema.

Tras todos los pasos realizados anteriormente podemos concluir algunos aspectos:

- Se han determinado los pasos y herramientas necesarios para realizar una inyección de fallos mediante un sistema software.
- Se ha completado un estudio del microprocesador Cortex-M1 enfocado en los aspectos de depuración así como de los sistemas de comunicación empleados en el sistema empotrado.
- Se ha conseguido realizar satisfactoriamente un manual de ayuda para la realización de sistemas embebidos con FPGA de Actel con un microprocesador Cortex-M1.
- Certeza del correcto funcionamiento del sistema empotrado conectado al PC y configuración satisfactoria de todos los parámetros hardware y software necesarios para realizar una inyección de fallos a través de un sistema software.
- Las pruebas experimentales muestran una alta tasa de acceso al microprocesador en depuración, siendo los registros de propósito general, los registros de configuración y la memoria accesibles tanto para lectura como para escritura. El único problema se ha encontrado en la lectura de los registros mapeados en memoria ya que el valor leído no correspondía con la realidad. Su escritura queda como interrogante al saber que el valor escrito ha sido correctamente trasmitido a los LEDs pero no por los registros que deberían hacerlo. Esto nos indica que no podemos realizar una campaña de inyección en ellos debido a que para alterar los registros en bits necesitamos leer su valor y realizarle posteriormente una operación lógica para modificar únicamente el bit deseado, y esto no ha sido posible. Sabemos además que la inviabilidad no es debido a la herramienta GDB ya que también hemos realizado numerosas pruebas en el software con el mismo resultado. Estos experimentos nos han llevado a pensar que la lectura de los diversos registros mapeados (los referidos a GPIO y UART concretamente) reciben un solo valor en todo su rango de

existencia en memoria, a pesar de la escritura en estas direcciones el valor de lectura es el mismo, lo cual nos hace pensar que los valores que obtenemos en lectura son una máscara que no refleja el estado real de los registros. No podemos decir con rigurosidad que este tipo de registro no sea accesible, lo que sí que podemos decir con seguridad es que el acceso de lectura a los registros no es posible ni en depuración ni por software ya que tampoco hemos podido acceder a ellos mediante la herramienta SoftConsole. Esto nos lleva a pensar que el problema podría radicar en el sistema creado para el estudio o en la placa de pruebas.

- Después de utilizar durante un número elevado de horas las herramientas para el diseño de hardware y software hemos encontrado alguna deficiencia en ambas:
    - La herramienta Libero Soc presenta un grado de inestabilidad elevado, ya que provoca numerosos fallos con el cambio de un PC a otro a la hora de ejecutar el programa.
    - Además la generación de información de bajo nivel no está suficientemente documentada, provocando dificultades a la hora de asignar variables entre la parte del software y el hardware.
    - La herramienta SoftConsole es muy completa en el campo de la depuración pero pensamos que tiene alguna deficiencia en el proceso previo a poder ejecutar un proyecto, ya que la asignación de lugares para las bibliotecas y la elección de un linker apropiado no resultan ni mucho menos obvias para un nivel usuario del programa, siendo muy complicada la solución de estos problemas.
    - La instalación de drivers para la comunicación de la placa con el ordenador resulta complicada en cuanto salimos del caso explicado en el tutorial de Actel (instalación desde el CD del programa), es decir, al descargar el software no disponemos de todas las carpetas que se necesitan en el tutorial por lo que debemos buscar software extra.

- Como última conclusión nos gustaría destacar que la relación entre el proyecto con el que hemos realizado las pruebas (proporcionado por Actel) y el proyecto que podemos ser capaces de diseñar siguiendo los tutoriales están a niveles muy distintos, ya que las bibliotecas disponibles en el software y las empleadas en el ejemplo son distintas.

6.2 Trabajos futuros

Observando las conclusiones obtenidas los trabajos futuros deberían ir enfocados en los siguientes aspectos:

- Investigación intensiva en la capacidad de lectura de registros mapeados en memoria ya que en la información proporcionada por el datasheet del microprocesador Cortex-M1 indica que esos registros han de poderse leer y escribir. La lectura ha sido inviable y la escritura queda en duda. Esto es debido al comportamiento anómalo de los registros de configuración del periférico GPIO, los cuales, teniendo funciones distintas cada uno de ellos, actuaban como registros de escritura. Por este motivo, alterando el valor de cualquier registro del GPIO la consecuencia era el cambio de valor de los LEDs, de manera que la escritura pero no de la manera esperada.
- Suponiendo que pudiéramos solucionar el punto anterior, el paso a seguir sería la creación de un script, o a mas nivel, un programa que permita configurar una inyección de fallos, pudiendo elegir las zonas afectadas, la gravedad de los fallos a inyectar y la capacidad de restaurar el programa para continuar con la inyección si algún error inyectado provoca que el programa pierda la secuencia de funcionamiento.
- Pensando en trabajos mas avanzados podríamos proponer una inyección real de fallos y comparar los resultados con la inyección programada a fin de asemejar

resultados y si los errores prácticos se alejan mucho de los teóricos cambiar el modo de funcionamiento del programa con el objetivo de tener una simulación teórica más parecida a la realidad.

Capitulo 7

Presupuesto

A continuación se detallan todos los gastos producidos a lo largo de la realización del proyecto, especificando el origen de estos y cuantificando detalladamente su fuente.

En este presupuesto tenemos en cuenta cuatro tipos de costes:

- Costes de personal: en estos costes se incluye el salario del ingeniero trabajador en el proyecto. La manera de cuantificar es mediante la variable "Hombre mes" que equivale a unas 131,25 horas trabajadas. Se estiman 400 horas trabajadas, lo que equivale a 3,05" hombres mes". El salario por "hombre mes" está estipulado en 2694,39 €, lo que suponen unos gastos de personal de 8217,79€.
- Coste de equipos: para cuantificar el coste de los equipos vamos a tener en cuenta cuarto aspectos siguiendo la siguiente fórmula:

$$\frac{A}{B} \text{xCxD}$$

- A: meses utilizados del equipo.
- B: vida total en meses del equipo.
- C: coste total del equipo.
- D: índice de utilización durante el proyecto (en %).

- Siguiendo esta fórmula se presentan los equipos empleados:

	A(mes)	B(mes)	C (€)	D (%)
FPGA M1AGL1000-DEV-KIT	6	24	500	70
Ordenador Sobremesa	6	60	700	100
Licencia Libero Soc	6	12	0	80

- De esta manera los costes por amortización de los equipos ascienden a 157,5 €
- Costes de transporte y dietas: estos costes hacen referencia los gastos producidos en gasolina y dietas realizados por el trabajador para reuniones o pruebas en laboratorio. Se cuantifican en 200€.
- Costes indirectos: estos costes hacen referencia a los gastos que se producen durante la realización del proyecto indirectamente, como puede ser el coste de la luz, agua o servicio de limpieza. Estos costes se estiman en un 20% del coste total. En nuestro caso ascienden a 1.715€.

A continuación se presenta la tabla resumen con el presupuesto detallando los cuatro aspectos mencionados anteriormente:

Presupuesto Costes Totales	**Presupuesto Costes Totales (€)**
Personal	8.218
Amortización	158
Transporte y dietas	200
Costes Indirectos	1.715
Total	10.290

Referencias

(1) http://www.actel.com/download/ Fuentes de descargas de Actel

(2) http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

Descarga de drivers para la comunicación con la placa.

(3) http://www.actel.com/techdocs/manuals/default.aspx. Manuales Actel

(4) http://www.actel.com/documents/M1IGLOO_DevKit_UG.pdf. Manual de instalación y ejemplos de la FPGA

(5) https://www.actel.com/support/default.aspx Ayuda de Actel